ISSN 0868-6157

Совместное советско-американское предприятие «СОВАМИНКО»

# КОМПЬЮТЕР ПРЕСС

## ОБОЗРЕНИЕ ЗАРУБЕЖНОЙ ПРЕССЫ



Средства организации многооконного интерфейса

В.Э.Фигурнов "IBM PC для пользователя"

2'91




поможет руководителям и инженерно-техническим работникам предприятий, совхозов и колхозов эффективно решить задачи управления производством, в том числе автоматизации учета и движения материальных ценностей.

Основу системы составляют микро-ЭВМ СМ 1810.62, СМ 1810.63, СМ 1810.64, СМ 1810.70, совместимые с IBM PC XT/AT.

*В СИСТЕМУ ВХОДЯТ:*

1. выносные интеллектуальные устройства связи с объектом, позволяющие автоматизировать любые типы предриятий;
2. широкий спектр печатающих устройств различного назначения;
3. локальная сеть, обеспечивающая подключение как отечественных, так и импортных ПЭВМ.

Главное преимущество системы ИКАР — это полная совместимость с пакетами прикладных программ, работающих в ОС MS DOS 3.30, наличие винчестера 20 Мбайт, сопроцессора, цветного монитора.

На основе технических средств системы ИКАР объединением разработаны различные программно-технические комплексы:

— учебные классы для школ и ПТУ с поставкой программного обеспечения для уроков информатики;

— информационные комплексы автоматизации учета и движения материальных ценностей в гостиницах, турбазах, санаториях, торговых, транспортных и промышленных предприятиях, колхозах, совхозах.

Использование специализированных печатающих устройств обеспечивает распечатку любых документов и автоматическую отрезку.

Объединение обеспечивает четырехлетний гарантийный ремонт поставленных вместе с системой технических средств и сопровождение программных продуктов.

Умеренные цены на выпускаемую продукцию: СМ 1810.63,64 — 17000 руб, СМ 1810.62 — 23000 руб., СМ 1810.70 — от 1000 до 8000 руб. делают наши системы вполне доступными для самого широкого круга пользователей.

*Телефоны для справок: (08600)3.12.10, (08600)3.84.98*

Совместное советско-американское предприятие «СОВАМИНКО»

# КОМПЬЮТЕР ПРЕСС

## ОБОЗРЕНИЕ ЗАРУБЕЖНОЙ ПРЕССЫ





2'91

Совместное советско-американское предприятие «СОВАМИНКО»

КОМПЬЮТЕР ПРЕСС

ОБОЗРЕНИЕ ЗАРУБЕЖНОЙ ПРЕССЫ





Адрес редакции:
113093, г.Москва, аб.ящик 37
Тел. для справок: 150-17-03
Бюро рекламы: 156-81-33
Факс: 200-22-89
E-mail:
postmaster@Computerpress.msk.su

# Дорогой читатель!

В сегодняшнем стремительно меняющемся мире разработка программного обеспечения и новых компьютерных технологий — это та область, в которой изменения происходят с поистине фантастической быстротой. Пользователь еще не успеет как следует познакомиться с новой версией какого-нибудь пакета, как на смену ему приходит следующая, новые процессоры, модели компьютеров и периферийных устройств бесконечной чередой сменяют друг друга, появляются новые компании, разоряются и исчезают старые.

Редакция журнала «КомпьютерПресс» стремится отслеживать все важнейшие изменения, происходящие в мире информатики, чтобы своевременно знакомить тебя с ними. Надеемся, что наше издание поможет тебе быть в курсе последних новостей и событий в области программного обеспечения и аппаратных разработок.

Наше издание — единственный в Советском Союзе ежемесячный журнал, посвященный этим проблемам.

Условия подписки, а также бланк заказа на получение журнала «КомпьютерПресс» наложенным платежом — на стр. 79 и 80.

*Годовая подписка на наш журнал —*

*это экономия вашего времени!*

Сдано в набор 5.02.91. Подписано к печати 14.02.91. Формат 84x108/16. Печать офсетная.
Усл.печ.л.8,4+0,32 (обл.). Тираж 100 000 экз. (1 завод–55 000). Заказ 2169 Цена 3 р. 15 к.
№025
Типография издательства «Калининградская правда»
236000, г.Калининград, ул.Карла Маркса, 18



*В прошлом номере журнала мы начали публикацию серии статей по архитектуре микропроцессоров фирмы Intel. Продолжая эту тему, авторы хотели бы подчеркнуть, что данные материалы никоим образом не претендуют на роль учебного пособия, являясь, скорее, популярными рассказами "из жизни мыслящих кристаллов".*

# Архитектура микропроцессоров

## Режимы адресации

Одной из основных функций центрального процессора любой ЭВМ является осуществление арифметических и логических операций над числами. Это возможно только после того, как процессору точно указаны места расположения этих чисел, или, согласно общепринятой терминологии, адреса операндов. Intel 80286, так же как и другие 16-разрядные процессоры, позволяет в командной строке указывать адреса двух операндов.

Однако все не так просто, как кажется на первый взгляд. Дело в том, что операнды могут находиться как в оперативной памяти, так и в регистрах самого процессора. Избежать путаницы в определении места их расположения помогают режимы адресации операндов, определенные архитектурой процессора. В нашем случае таких режимов — четыре: непосредственный, регистровый, а также прямой и косвенной адресации памяти.

Непосредственный режим позволяет в качестве операнда в строке программы, написанной на ассемблере, указывать само число, над которым необходимо произвести действие. Применение этого режима имеет свои достоинства и недостатки: с одной стороны, процессор получает данные быстрее, чем при обращении к ОЗУ, а с другой, это перегружает текст программы и к тому же, при неоднократном использовании программой одних и тех же операндов, их удобнее хранить в ОЗУ или в регистрах.

Регистровый режим дает возможность указывать в программной строке имя того регистра, в котором в настоящий момент находится операнд. Ввиду того, что содержимое регистра по самым разным причинам в процессе выполнения программы может изменяться, действия можно производить как над константами, так и над переменными или элементами массивов при помощи одинаковых программных строк.

В случае нахождения операнда в ОЗУ, его адрес можно хранить в регистрах, причем в команде указывается, например, имя регистра, в котором находится адрес сегмента, а смещение дается в явном виде. Допускается хранение в регистрах и адреса сегмента, и смещения; при этом, в команде указываются только их имена. Подобный способ называется косвенной адресацией памяти. Он позволяет "находить" необходимые операнды с использованием комбинаций, в которых могут участвовать: базовый регистр, индексный регистр, базовый регистр+индексный регистр, базовый регистр+смещение, индексный регистр+смещение, базовый регистр+индексный регистр+смещение.

Кроме того, и сегмент, и смещение могут быть указаны непосредственно в команде. Такой способ называется прямой адресацией памяти и зачастую применяется при обращении к фиксированной ячейке какого-либо сегмента.

Операндами являются как 8-, так и 16-разрядные числа. Для того, чтобы процессор не перепутал, с какими данными ему предстоит иметь дело и не "промахнулся" при обращении к оперативной памяти, в формате команды предусмотрен бит, сигнализирующий, операнды какой длины должны быть обработаны.

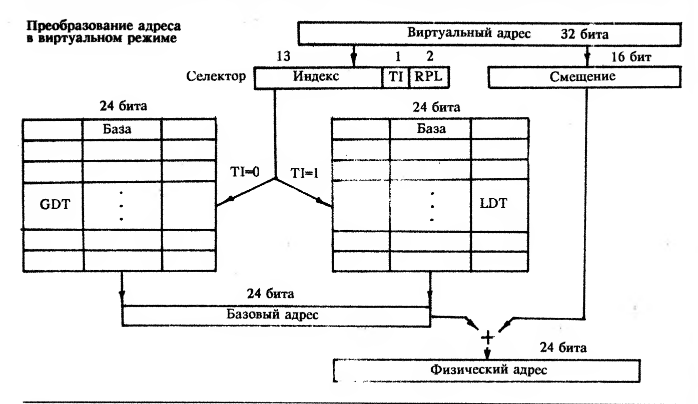
### Режим виртуального адреса

В предыдущем выпуске мы рассмотрели режим реального адреса, позволяющий обращаться к ОЗУ объемом до одного мегабайта при ширине адресной шины процессора 80286 всего 16 разрядов. Но каждому, кто когда-либо брал в руки журналы по компьютерной технике, наверняка попадались на глаза статьи, в которых черным по белому было написано, что оперативная память PC/AT может иметь размеры четыре, а то и восемь мегабайт. Подобные цифры вызывают законное недоумение. Во-первых, непонятно, зачем персональному компьютеру такое ОЗУ, а во-вторых, каким образом им пользоваться, если для адресации одного только мегабайта приходится применять довольно хитроумный двухступенчатый способ.

На первый вопрос ответить сравнительно просто. Увеличение объема ОЗУ объясняется, в основном, возможностью применения на персональных компьютерах многозадачных операционных систем, работающих в режиме разделения времени — таких, как, например, OS/2 и UNIX. В двух словах принцип разделения времени состоит в следующем: время работы процессора искусственно квантуется таким образом, что в течение каждого кванта, составляющего долю секунды, выполняется "кусок" одной из программ, одновременно запущенных на компьютере; затем выполняется "кусок" из другой программы, и так далее. Разумеется, для хранения текстов программ, их данных и промежуточных результатов использовать такие сравнительно медленные запоминающие устройства, как жесткие диски, не имеет смысла — полностью теряется эффект одновременности выполнения сразу нескольких задач. Таким образом, увеличение пространства ОЗУ — вынужденная мера, направленная на повышение производительности компьютера.

Что же касается адресации оперативной памяти "за границей" одного мегабайта, то здесь разработчикам i286 помог принцип виртуальной организации памяти.

Как и в случае реального адреса, результирующий физический адрес памяти является суммой некоторой базы и смещения. Со смещением нам, в общем, уже все ясно, а вот базовый адрес определяется сложнее. Сегментный регистр содержит не само число, из которого путем добавления четырех нулей справа получается адрес сегмента, а некоторый идентификатор, называемый селектором. Он состоит из трех полей (см.рисунок), два из которых — индикатор TI и индекс — определяют адрес искомого сегмента по специальным таблицам, находящимся в ОЗУ. Каждый элемент любой из этих таблиц, называемый дескриптором, имеет длину 64 разряда, 24 из них отведены под базовый адрес, а остальные для нас пока интереса не представляют. Если при вычислении адреса индикатор



селектора TI равен нулю, то процессор обращается к так называемой глобальной дескрипторной таблице, которая может использоваться всеми запущенными задачами. В противном случае (при TI = 1) используется одна из локальных дескрипторных таблиц. Дело в том, что если глобальная дескрипторная таблица (GDT) в системе всего одна, то локальная (LDT) — для каждой из задач своя. Сумма смещения и базового адреса, взятого из дескрипторной таблицы, дает в результате искомый физический адрес в ОЗУ (см.рисунок). Длина адреса в таком случае составляет целых 24 бита, а это позволяет обращаться уже к 16 Мбайтам оперативной памяти.

Итак, аппаратно-реализуемое адресное пространство i286 составляет 16 Мбайт, но стоит заметить, что длина индекса в селекторе равна 13 разрядам, а размер сегмента равен $2^{16}$ байтам. Таким образом, каждая задача, используя свою LDT, получает адресное пространство $2^{29}$ байт, а в сумме с областью, определяемой GDT, эта величина составляет $2^{30}$ байт или 1 Гбайт.

Теперь у читателя может возникнуть вполне резонный вопрос: зачем так много? Ответим: для реализации многозадачного режима. При недостаточно больших размерах ОЗУ можно организовать доступ к внешним запоминающим устройствам, как к некой области виртуальной памяти. Тогда в ОЗУ достаточно держать лишь наиболее часто запрашиваемые сегменты задачи, а остальные "подкачивать" с жесткого диска по мере необходимости.

### Прерывания

Часто возникают ситуации, когда вам необходимо ввести данные при помощи мыши или с клавиатуры, а процессор занят выполнением какой-то задачи. Как быть? Или другой пример: вечером вы спокойно работаете в опустевшем здании НИИ, а в это самое время ничего не подозревающий вахтер в целях пожарной безопасности обесточивает все помещения. Каким образом организовать работу процессора, чтобы он мог предпринимать какие-то действия в ответ на подобные внешние раздражители?

Решить эту проблему можно двумя способами. Первый, более "древний", заключается в том, что процессор со строго определенной частотой опрашивает остальные устройства системы, которые в свою очередь могут послать ему в ответ некий признак своей активности. Тогда процессор по специальному алгоритму приостанавливает ход выполняемой программы и обслуживает активное внешнее устройство. К сожалению, такой метод существенно снижает производительность, так как опросы происходят независимо от готовности внешних устройств начать взаимодействие с центральным процессором.

Другой метод построен на использовании прерываний от внешних устройств. Прерывания — это посылаемые устройствами процессору специальные сигналы о необходимости передачи какой-то информации, либо о сбойной ситуации. Систему прерываний можно организовать по-разному. Известен, например, такой путь: в зависимости от "важности" прерываниям присваиваются приоритеты разного уровня, причем самым низким приоритетом "награждается" центральный процессор. И действительно, если дать ему наивысший приоритет, он будет напоминать большого начальника, который заперся в своем кабинете и не отвечает на звонки подчиненных.

В случае i286 система прерываний построена несколько иначе. Соответствующие сигналы от устройств, входящих в вычислительную систему, могут поступать на один из двух входов процессора: NMI или INTR, причем вход NMI обладает безусловным приоритетом, так как сюда поступают сигналы только о катастрофических событиях, например, в случае неожиданного отключения питания. На вход INTR поступают прерывания не столь значительные, и процессор может их проигнорировать, если он в это время занят чем-то более важным. О состоянии большой занятости сигнализирует флажок разрешения прерываний IF: если он равен нулю, то процессор на "звонки" не отвечает.

Посмотрим, что же происходит, когда IF = 1 и на вход INTR поступил сигнал прерывания. Сначала процессор приостанавливает выполнение текущей программы и запоминает промежуточные результаты в стеке, необходимые для восстановления статус кво. Затем он посылает устройству-возмутителю спокойствия запрос о причинах прерывания. В ответ может быть получено некоторое число в диапазоне от 0 до 255 — такое количество вариантов сообщения внешнего устройства разработчики предусмотрели для указания процессору, какие шаги следует предпринять в каждом конкретном случае. Число от 0 до 255 определяет тип прерывания, а инструкции о том, как процессору реагировать на тот или иной тип, содержатся в специальных программах, называемых процедурами прерываний. Адреса процедур находятся в специальной таблице, состоящей, как вы уже догадались, из 256 элементов, причем каждый элемент содержит соответствующие значения регистров IP и CS. Начальный адрес таблицы в ОЗУ — 0. По окончании процедуры прерывания процессор возвращает из стека промежуточные данные и продолжает выполнять прерванную программу с точки прерывания.

Если сигнал поступает на вход NMI, процессор не запрашивает у устройства информацию о типе прерывания, так как его причиной может стать только катастрофическая ситуация, и у процессора в этом случае всего одна задача — спасать наиболее важные результаты в безопасное место. Подобная ситуация в таблице соответствует прерыванию второго типа.

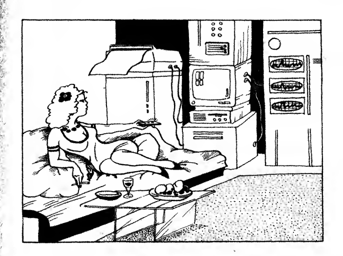
Кроме прерываний от внешних устройств, возможны и внутренние прерывания, возникающие в самом процессоре. Такие сигналы могут генерироваться, например, при разнообразных ошибках во время выполнения программ, а также для организации пошагового режима работы процессора (см.таблицу).

**Зарезервированные прерывания**

| | Реальный режим | Виртуальный режим |
|---|---|---|
| Номер | Прерывание | Прерывание |
| 0 | Особый случай ошибки деления | Особый случай ошибки деления |
| 1 | Прерывание пошаговой работы | Прерывание пошаговой работы |
| 2 | Немаскируемое прерывание | Немаскируемое прерывание |
| 3 | Контрольный останов | Контрольный останов |
| 4 | Особый случай переполнения | Особый случай переполнения |
| 5 | Особый случай превышения диапазона | Особый случай превышения диапазона |
| 6 | Особый случай недействительного кода операции | Особый случай недействительного кода операции |
| 7 | Особый случай отсутствия сопроцессора | Особый случай отсутствия сопроцессора |
| 8 | Особый случай слишком малой IDT | Особый случай слишком малой IDT |
| 9 | Особый случай превышения сегмента сопроцессором | Превышение сегмента сопроцессором |
| 10 | Зарезервировано | Недействительный TSS |
| 11 | Зарезервировано | Отсутствие сегмента |
| 12 | Зарезервировано | Особый случай стека |
| 13 | Особый случай превышения сегмента | Особый случай защиты |
| 14 | Зарезервировано | Зарезервировано |
| 15 | Зарезервировано | Зарезервировано |
| 16 | Особый случай сопроцессора | Особый случай сопроцессора |
| . | | |
| . | Зарезервированы | Зарезервированы |
| . | | |
| 31 | | |

*(Продолжение следует)*

*И.Липкин*



*В настоящее время в мире эксплуатируется около 500000 рабочих станций. Однако, по прогнозам экспертов, в 1993 г. объем продажи рабочих станций только в странах Западной Европы превысит эту величину и в 1994 г. будет эксплуатироваться свыше 2 млн. рабочих станций.*

# О СОСТОЯНИИ И ПЕРСПЕКТИВАХ РАБОЧИХ СТАНЦИЙ

Эволюция рабочих станций и тенденции развития рынка их сбыта во многом повторяют путь, по которому осуществлялось развитие в области производства и сбыта персональных компьютеров. Наблюдается стремительный рост объема продажи рабочих станций как в стоимостном, так и в натуральном исчислении. Если в 1982 г. мировой объем продажи составлял 25 млн. долл., то в 1988 г. он превысил 4 млрд. долл. По оценкам экспертов фирм Dataquest и Frost & Sullivan, специализирующихся в области исследований рынка сбыта средств вычислительной техники, в 1992 г. эта цифра должна превысить 20 млрд. долл. Рост объема продажи в натуральном исчислении превышает его рост в стоимостном исчислении, причем преобладание этой тенденции свидетельствует об уменьшении средней стоимости вычислительных систем данного класса.

В настоящее время основной объем продажи рабочих станций (86%) приходится на долю четырех фирм: Hewlett-Packard, Apollo Computers (эта фирма недавно была приобретена фирмой Hewlett-Packard), Sun Microsystems и DEC. Однако в ближайшее время конкуренция на рынке сбыта сильно обострится, что связано с появлением новых изделий фирм IBM, Acorn, Sun Microsystems, Compaq и Hewlett-Packard (эти изделия будут более подробно описаны в заключительной части настоящего обзора).

## *АППАРАТНЫЕ СРЕДСТВА РАБОЧИХ СТАНЦИЙ*

Основной сферой применения рабочих станций являются области, требующие осуществления большого количества вычислений и управления высококачественной графикой. К этим областям относятся, прежде всего, разнообразные системы автоматизации инженерного труда и автоматизированного проектирования (в первую очередь, в области конструирования изделий электронной техники), системы моделирования и анализа данных. Вторую группу проблем, для решения которых применяются рабочие станции, составляют вопросы автоматизированной разработки программного обеспечения.

Круг задач, решаемых с помощью рабочих станций, определяет предъявляемые к ним требования. Например, при использовании машины для решения задач цифрового моделирования превалирующими являются требования к быстроте выполнения целочисленных операций. При моделировании матриц интегральных схем предпочтительно размещать всю информацию в ОЗУ, так как в противном случае частое обращение к дисковым ЗУ будет замедлять работу программы. При

моделировании аналоговых характеристик необходимо, чтобы операции с плавающей запятой выполнялись как можно быстрее. А при работе с базой данных более важным моментом является скорость обращения к дисковым ЗУ.

Большое значение имеет и быстродействие при обработке графической информации. Адекватное отображение сложных графических образов важно не только в научно-исследовательских приложениях, но и при размещении элементов на печатных платах или в интегральных схемах. В работе систем автоматизации инженерного труда важна также способность рабочей станции быстро воспроизводить изменившееся графическое изображение. Обязательным требованием к машине сегодня является возможность управлять видеомонитором с экраном высокого разрешения. Отсюда вытекает необходимость того, чтобы емкость ОЗУ составляла не менее 4 Мбайт.

С 1989 г. в качестве центрального процессора на рабочих станциях используются микропроцессоры с RISC-архитектурой (с сокращенным набором команд). В отличие от процессоров с CISC-архитектурой (со сложным набором команд), для управления работой микропроцессора с RISC-архитектурой применяется сравнительно небольшое число простых команд, набор которых оптимизирован. Поскольку время выполнения этих простых команд невелико, скорость работы микропроцессора, а, следовательно, и быстродействие машины существенно повышается.

В настоящее время на рынке широко представлены рабочие станции на базе микропроцессоров как с CISC-, так и с RISC-архитектурой (см. рис.1). Это же относится и к новейшим системам, которые только выходят на рынок: фирмы Hewlett-Packard и Compaq используют процессоры с CISC-архитектурой, а фирмы IBM, Sony и Acorn — процессоры с RISC-архитектурой. Как правило, машины на базе микропроцессоров с CISC-архитектурой дешевле, чем рабочие станции на базе микропроцессоров с RISC-архитектурой, и для них существует больший выбор программного обеспечения. Однако, быстродействие микропроцессоров с сокращенным набором команд почти вдвое выше.

Относительная дешевизна рабочих станций на базе микропроцессоров со сложным набором команд объясняется тем, что комплектующие их изделия более универсальны и количество этих изделий меньше. Для изготовления машин на базе микропроцессоров с RISC-архитектурой требуются быстродействующие, а следовательно, дорогие схемы памяти. Кроме того, объем оперативной памяти должен быть достаточно большим (минимум 8 Мбайт).

Умеренная цена некоторых машин объясняется еще и тем, что их системные платы не имеют расширительных разъемов. Все внешние устройства подключа-

**Процессор вычислительной системы**

АРМ с RISC-архитектурой

Data General AViiON
Opus 18000 88000
Tektronix XD 88/10

DEC station 2100
3100 MIPS
MIPS RS 2030 R2000; 3000
Silicon Graphics Personal Iris

HP 834CH HP-PA
Inegraph Interpro 125 Clipper
Sun SPARC station
Solbourne 4/500 Sun SPARC

Высокое быстродействие
Небольшой объем ПО

АРМ с CISC-архитектурой

Apollo 2500
3500
4500 68030
Sony 1700
Sun 3180

Sun 386 80386

Большой объем ПО
Низкая стоимость
Невысокое быстродействие

**Рис.1. Используемые микропроцессоры и некоторые особенности вычислительных систем.**

**Архитектура вычислительной сети**

| С центральным процессором | С центральным процессором и дополнительными подсистемами ускорения работы графики | С распределенными процессорами |
|---|---|---|
| DECstation 2100<br>3100<br>Data General AViiON<br>MIPS RS 2030<br>Apollo 2500 | Sun SPARCstation 1<br>Sun 3/80<br>Apollo 3500<br>4500<br>Sony 1700<br>Silicon General Personal Iris | Opus 1800: 88000 — это ЦП, процессор 80386 для AT-шины системы ввода/вывода<br><br>Integraph Interpro 125 — Clipper — это ЦП; 80186 — в качестве процессора системы ввода/вывода; спецпроцессор для графики<br><br>Tektronix XD 88/10: 88000 — это ЦП, специализированный процессор графики |
| Простое РО и технические средства<br>Низкая стоимость | Использование акселераторов графики может увеличить стоимость АРМ вдвое | ПО более сложное<br>Процессор для системы ввода/вывода часто является "узким местом" |

**Рис.2. Архитектура некоторых вычислительных систем и связанные с ней особенности**

ются через SCSI-интерфейс. По этому пути пошла, например, фирма DEC, в своей системе DECstation 2100. В отличие от фирмы DEC фирма Sun Microsystems использует для моделей SPARCstation 1 фирменную шину расширения SBus, которая необходима для подключения дополнительного периферийного оборудования. Этим, в частности, можно объяснить то, что при почти одинаковых возможностях рабочая станция фирмы Sun Microsystems несколько дороже.

Стоимость машины во многом зависит от архитектуры вычислительной системы. Применение графических акселераторов и спецпроцессоров системы ввода-вывода улучшает эксплуатационные характеристики, но ведет к удорожанию рабочих станций. На рис.2 показаны некоторые особенности архитектуры имеющихся на рынке машин.

Основные технические характеристики и ориентировочная стоимость ряда сравнительно недорогих рабочих станций представлены в приведенной ниже таблице. Оценивая их стоимость, следует иметь в виду, что итоговая цена комплекта в значительной степени зависит от общего объема ОЗУ, количества и емкости дисковых ЗУ и качества видеомонитора.

## *НЕКОТОРЫЕ АСПЕКТЫ СТАНДАРТИЗАЦИИ И РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ РАБОЧИХ СТАНЦИЙ*

Хотя сегодня практически все рабочие станции функционируют или могут функционировать под управлением различных версий ОС UNIX, машины различных фирм остаются практически несовместимыми на уровне программного обеспечения. Поэтому прикладные программы для вычислительных средств этого вида остаются дорогими. То же самое можно сказать и о дополнительном периферийном оборудовании, поскольку каждый его вид должен разрабатываться для одной серии рабочих станций.

Проблема выбора стандарта и обеспечения совместимости остается весьма острой. Стандартизация очень выгодна пользователям, так как в этом случае и пакеты прикладных программ, и дополнительное оборудование становятся гораздо дешевле. В настоящее время процесс стандартизации и обеспечения совместимости рабочих станций осложняется обострением

2 Зак. 2169

конкурентной борьбы и отсутствием явного лидера, так как каждой ведущей фирме-изготовителю рабочих станций хотелось бы, чтобы процессор и архитектура, используемые именно в ее машине, легли в основу будущего стандарта для этого класса вычислительных систем.

Так, фирма Sun Microsystems адаптировала программное обеспечение, разработанное для ранних моделей ее рабочих станций на базе процессоров Motorola 68030 и Intel 80386, для использования на рабочей станции SPARCstation 1 на базе микропроцессора с RISC-архитектурой SPARC (Scalable Processor Architecture — масштабируемая процессорная архитектура). Она продала лицензии на изготовление этого процессора фирмам Fujitsu и Texas Instruments. Компания Solbourne Computers приобрела у Sun Microsystems лицензию не только на процессор SPARC, но и на программное обеспечение для этого процессора — ОС SunOS UNIX, программный пакет Sun View, систему управления окнами NeWS, сетевую файловую систему Network File System (NFS) и управляющую систему Open Network Computing (ONC), которые широко используются в выпускаемых этой фирмой рабочих станциях. В ближайшее время на рынке ожидается появление машин на базе микропроцессора SPARC, выпускаемых фирмами Toshiba и Tatung.

Фирмы, производящие рабочие станции на базе RISC-микропроцессора Motorola 88000, такие как Tektronix, Data General и Opus, надеются, что большое количество нужных для рабочих станций прикладных программ будет создано благодаря усилиям консорциума, названного 88open. Этот консорциум был создан для того, чтобы способствовать принятию стандартов и формированию процедур сертификации программного обеспечения для вычислительных систем на базе RISC-микропроцессора Motorola 88000. Выполнение этих задач позволило бы создать открытую среду для программного обеспечения подобных систем. Подкомитет консорциума, названный 88open Binary Compatibility Standard (Стандарт двоичной совместимости), разработал описание стандартной формы всех исполняемых программ для вычислительных систем на базе микропроцессора Motorola 88000. В стандарте затронуты вопросы представления данных программ и файлов. В нем определены часто встречающиеся типы данных (целые, с плавающей запятой и символьные) и указывается, как данные располагаются в слове и на странице памяти и каким образом файл размещается на диске.

Любая программа, разработанная сторонним поставщиком, может работать на любой машине, выпускаемой фирмой, входящей в консорциум. Поэтому темпы увеличения объема программного обеспечения для этих рабочих станций будут расти.

## *НОВОЕ ПОКОЛЕНИЕ РАБОЧИХ СТАНЦИЙ*

Как уже отмечалось, широкое применение новых микропроцессоров с повышенным быстродействием и отсутствие явного лидера в сфере производства рабочих станций привело к появлению на рынке новых моделей как на базе старших моделей ПЭВМ, так и на базе новых микропроцессоров с RISC-архитектурой. Ниже мы приводим более подробную информацию о некоторых представителях нового поколения рабочих станций.

### Рабочая станция RS/6000 POWERstation 320 Фирма IBM

*Технические характеристики*
Фирма-изготовитель: IBM.
Наименование: RS/6000 POWERstation 320.
Процессор: SGR2032, 20 МГц.
ОЗУ: 8 Мбайт с расширением до 32 Мбайт.

Порты: один параллельный, два последовательных, соединители для клавиатуры и манипулятора типа "мышь".

Накопители: стандартно 89-мм НГМД и ЗУПД емкостью 120 Мбайт (подробнее см. текст ниже).

В настоящее время фирме IBM принадлежит менее 3% рынка сбыта рабочих станций и ее машины серии 6150 успеха на рынке не имеют. Однако, выпуская на рынок новое семейство машин RISC System/6000, фирма IBM рассчитывает завоевать от 15 до 20% рынка сбыта рабочих станций.

Младшим представителем этого семейства является рабочая станция RS/6000 POWERstation 320. POWER — сокращенное название новой RISC-архитектуры — Performance Optimisation With Enchanced RISC (оптимизация производительности с усовершенствованной RISC-архитектурой).

Эта машина выпускается в двух вариантах — напольном (466 мм x 280 мм x 523 мм, вес 15,4 кг) и настольном (162 мм x 456 мм x 523 мм, вес 12,7кг).

Внешний вид POWERstation 320 в настольном исполнении напоминает ПЭВМ PS/2 Model 70. В верхней части передней панели располагается скважина для ключа, выключатель питания, клавиша перезапуска и 89-мм НГМД. На задней панели размещаются четыре MCA-гнезда, один параллельный и два последовательных порта, а также разъемы для подключения клавиатуры и манипулятора типа "мышь".

Расширение производится с помощью шины Micro Channel с увеличенной пропускной способностью, которая будет принимать все платы MCA. Для более эффективного использования производительности нового процессора пропускная способность этой шины повышена до 40 Мбайт/сек.

Стандартная модель POWERstation 320 комплектуется жестким диском (или, выражаясь языком фирмы IBM, DASD — Direct Access Storage Device — ЗУ прямого доступа) емкостью 120 Мбайт и ОЗУ объемом 8 Мбайт; 52-разрядная адресация дает гигантский с теоретической точки зрения размер виртуальной памяти — 256 Тбайт. На практике POWERstation 320 ограничена ОЗУ объемом 32 Мбайта и НМД емкостью 640 Мбайт. Предусмотрена возможность установки двух контроллеров SCSI (один такой контроллер является стандартным в старших моделях рабочей станции POWERstation).

Диапазон возможных накопителей охватывает 133-мм НГМД емкостью 1,2 Мбайта, внешний портативный НМД емкостью 355 Мбайт или 670 Мбайт, малогабаритный внешний НМД емкостью 320 Мбайт, стример на 150 Мбайт (ширина ленты 6,35 мм), систему НМЛ емкостью 2,3 Гбайта (ширина ленты 8 мм), внешний дисковод для ПЗУ на компакт-дисках и 9-дорожечный накопитель на магнитной ленте шириной 12,7 мм.

Широкий ассортимент новых и существующих МСА-плат обеспечивает соединение не только с локальной сетью Token Ring, но и с Ethernet, X.25 и 3270.

Используемый в POWERstation 320 процессор SGR2032, работающий в этой модели с тактовой частотой 20 МГц, является набором из девяти кристаллов. В этот набор, который в значительной степени был спроектирован для работы с ОС AIX, входят отдельные математические процессоры с фиксированной и плавающей запятой и процессор передачи управления.

Процессор передачи управления за один такт выполняет команду регистра логического условия и команду перехода и обеспечивает выбор всех команд. Некоторые из них выполняются этим же процессором, однако большинство команд передается процессорам с фиксированной и плавающей запятой. После передачи арифметической команды процессор передачи управления может выбрать следующую несмотря на то, что арифметический процессор все еще занят.

Процессор с фиксированной запятой содержит тридцать два 32-разрядных регистра общего назначения и выполняет за один такт одну операцию. Процессор с плавающей запятой может параллельно выполнить за один такт две 64-разрядных операции — "умножение и сложение". Конвейер потока данных процессора с плавающей запятой организован так, что этот процессор может в каждом такте инициировать новую команду "умножить и сложить".

Использование кэш-памяти данных емкостью 32 Кбайта, кэш-памяти команд емкостью 8 Кбайт и линии связи, позволяющей передавать до четырех команд, дает возможность выполнять за один такт пять операций. По данным, представленным фирмой IBM, базовая машина обладает феноменальным быстродействием — 27,5 млн. команд/сек и 7,4 млн. операций с плавающей запятой/сек (для сравнения отметим, что для описанных в обозрении КомпьютерПресс № 8, 1990 г. рабочих станций DECstation 3100 фирмы DEC и SPARCstation I фирмы Sun Microsystems заявлены следующие характеристики: 14,3 и 12,5 млн. команд/сек и 1,6 и 1,4 млн. операций с плавающей запятой/сек, соответственно).

Для того, чтобы в полной мере использовать быстродействие нового процессора, рабочие станции семейства RS/6000 могут быть оснащены новыми НМД, 89-мм с емкостью 320 Мбайт и 133-мм с емкостью 857 Мбайт, характеризующимися временем доступа 12,5 и 11,2 мсек и скоростью передачи данных 2 Мбит/сек и 3 Мбит/сек, соответственно.

Эти машины работают под управлением новой версии ОС UNIX фирмы IBM — AIX (Advanced Interactive eXecutive) Version 3. На уровне исходных кодов эта ОС совместима с Base System V Interface Definition (SVID) и BSD 4.3 фирмы AT&T. AIX поддерживает POSIX (Portable Operating System for Computer Environments), TCP/IP и Network File System (NFS), а также языки программирования Си, ФОРТРАН, КОБОЛ и Паскаль и множество прикладных программ, функционирующих под управлением ОС UNIX.

Удобство в работе с ОС AIX в настоящее время определяется применением двух пакетов. Первый пакет известен как AIXwindows Environments/6000. Он основан на улучшенной версии OSF Motif и является частью пакета, включающего X Window System, OSF/Motif и AIXwindows Desktop. Альтернатива, которая появится в ближайшее время, известна как AIX Graphic User Environment/6000 (графическая среда пользователя). Она в значительной степени основывается на объектно-ориентированной среде NeXTStep. Обе системы обладают возможностями Display PostScript, а AIXwindows поддерживает AIX X-GSL (Graphic Subroutine Library — библиотека графических подпрограмм) фирмы IBM и GL фирмы Silicon Graphics.

Необычным новшеством является то, что привычная объемная документация по ОС UNIX заменена на ПЗУ на компакт-диске (CD-ROM). К документации обращаются с помощью текстовой и графической системы InfoExplorer, обеспечивающей быструю выборку информации и наведение перекрестных справок.

Цена POWERstation 320 во многом зависит от комплектации системы. Система в минимальной (базовой) конфигурации, в которую входят ОЗУ объемом 8 Мбайт, НМД емкостью 120 Мбайт, клавиатура, манипулятор типа "мышь", адаптер сети Ethernet, ОС AIX, среда интерфейса пользователя, NFS, система поиска и выборки информации для работы с документацией и видеомонитор, который, честно говоря, является чисто текстовым, стоит около 6400 ф.ст. Следующий вариант этой модели оснащается 48-см (19-дюймовым) полутоновым видеомонитором с разрешением 1240x1024 и платой Greyscale Graphics Display Adaptor (адаптер полутонового графического дисплея), обеспечивающей отображение 16 оттенков серого из палитры в 256 оттенков. Стоимость этого варианта POWERstation 320

приближается к 10400 ф.ст. Оснащение рабочей станции удвоенным НМД и замена полутонового видеомонитора на цветной с таким же экраном и разрешением повышает цену системы почти до 14700 ф.ст.

## Рабочая станция News 3860
## Фирма Sony Microsystems

*Технические характеристики*

Фирма-изготовитель: Sony Microsystems.

Наименование: News 3860/5.

Процессор: MIPS R3000RISC с частотой 16,67 МГц, арифметический сопроцессор с плавающей запятой R3010.

ОЗУ: 16 Мбайт стандартно.

Порты: два последовательных, сетевой адаптер Ethernet, порты SCSI, клавиатуры и манипулятора типа "мышь", адаптер монохромного видеомонитора.

Накопители: 89-мм НГМД емкостью 1,44 Мбайта, НМД емкостью 640 Мбайт.

Выпустив свыше 5000 рабочих станций на базе микропроцессора Motorola 680X0, фирма Sony Microsystems перешла на выпуск машин News 3860/5 на базе микропроцессора с RISC-архитектурой. Интересно отметить, что целый ряд компонентов, используемых в них, импортирован из США.

Внешний вид News 3860/5, в корпусе которой (533 мм x 711 мм x 508 мм, вес 24,5 кг) размещаются НГМД и НМД, напоминает ПЭВМ класса IBM PC/AT.

В News 3860 используется процессор MIPS R3000, работающий с тактовой частотой около 17 МГц. Для работы в режиме нулевого ожидания предусмотрены 64-килобайтная кэш-память данных и 64-килобайтная кэш-память команд. Архитектура рабочей станции рассчитана на поддержку перспективных ИС MIPS, которые будут обладать гораздо большим быстродействием, чем используемый в настоящее время процессор R3000. В качестве арифметического сопроцессора используется стандартный процессор с плавающей запятой MIPS R3010, дающий 32-разрядную одинарную точность и 64-разрядную удвоенную точность.

От других рабочих станций, в которых используется процессор MIPS, машины фирмы Sony Microsystems отличаются тем, что используют дополнительные процессоры 68030 для улучшения производительности системы ввода-вывода. Ведутся работы по применению дополнительных процессоров 68030 в качестве графических и сетевых процессоров. Каждый из этих процессоров будет поставляться на собственной плате расширения, подключаемой к фирменной шине Sony, которая поддерживает 7 гнезд и работает со скоростью 128 Мбайт/сек. Каждый процессор, каждая плата памяти и платы цветной графики занимают по одному гнезду этой шины, которая соответствует стандартам ECC.

Безусловно, рабочая станция News 3860/5 не является многопроцессорной системой в истинном смысле слова, так как дополнительный процессор лишь в некоторой степени снижает нагрузку на основной процессор. Мощность данной системы максимально реализуется в однопользовательских прикладных задачах с интенсивными операциями ввода-вывода.

Фирма Sony характеризует быстродействие News 3860 как 20 млн. команд/сек и 3,5 млн. операций с плавающей запятой/сек.

ОЗУ стандартной рабочей станции имеет объем 16 Мбайт и составлено из 1-Мбит ИС, смонтированных на плате, занимающей одно гнездо. При наличии достаточного количества свободных гнезд максимальный объем ОЗУ может быть доведен до 80 Мбайт. Фирма Sony надеется в будущем довести объем памяти, размещаемой на одной плате, до 64 Мбайт, используя 4-Мбит ИС. Поскольку внутренняя фирменная шина не имеет внешнего порта, использование внешней памяти невозможно, а все платы памяти необходимо приобретать у фирмы Sony. Единственной большой заказной ИС, которая разработана фирмой Sony, является блок управления памятью, хотя эта ИС полностью соответствует спецификации фирмы MIPS.

В корпусе настольной рабочей станции News 3860/5 можно разместить только один винчестер емкостью 640 Мбайт. Можно добавить подсистему SCSI, позволяющую подключать дополнительные ЗУ емкостью до 981 Мбайта. В корпусе имеется место для установки факультативного НМЛ емкостью 150 Мбайт, который может также читать ленты более старого типа емкостью 80 Мбайт. НГМД представлен 89-мм накопителем емкостью 1,44 Мбайта.

Монохромная графика с разрешением 1152x900 элементов изображения поддерживается на основной плате; использование единственной платы расширения позволяет добавить цветность. Фирма Sony использует наиболее популярную систему цветной графики, функционально идентичную стандарту Sun CG4 — 8-бит и два наложения. Это позволяет выводить на экран до 256 цветов из палитры в 16,8 млн. Хотя к фирменной шине и подключена шина VME, она обладает далеко не всеми возможностями стандартной VME-шины. Требование подключения графических плат к фирменной шине ведет к тому, что стандартные адаптеры улучшенной графики, поставляемые в виде VME-плат, не будут работать на этой системе. Используемые видеомониторы Sony Trinitron, как 48-см цветной, так и 43-см монохромный, имеют великолепные характеристики.

На задней панели News 3860/5 смонтированы два последовательных порта, адаптер сети Ethernet, порт SCSI, адаптер монохромного видеомонитора, порты для подключения клавиатуры и манипулятора типа "мышь".

В настоящее время в качестве операционной системы на рабочей станции News 3860/5 применяется AT&T UNIX System V версия 4.1. Для того, чтобы использовать весь потенциал своей архитектуры с про-

цессорами MIPS и 68030, фирма переписала ядро AT&T. Для News 3860 можно использовать практически все дополнительное программное обеспечение AT&T System (по лицензии), включая интерфейс OSF Motif, NFS, NeWS, Xterm и Kterm, DBX и XDBtools, GKS/C, FIGARO, Си и ФОРТРАН 77. У фирмы MIT Sony приобрела лицензию на XWindows.

К концу 1990 г. компания Sony рассчитывала предложить для рабочих станций серии News 3860 около 1000 прикладных программ. Фирма предназначает свои рабочие станции для решения, прежде всего, задач автоматического проектирования (MCAD и ECAD), обработки изображения и автоматизации разработки программного обеспечения.

Что же касается цены, то базовая бездисковая система с одним процессором, цветной графикой и ОЗУ объемом 16 Мбайт стоит около 9500 ф.ст. Стоимость такой же системы, оборудованной НМД и НМЛ, достигает почти 13500 ф.ст. Для сравнения отметим, что стоимость системы News 3860/8 (самая старшая модель, предназначенная, в первую очередь, для использования в качестве файл-сервера сети) с ОЗУ объемом 32 Мбайта, НМД емкостью 1,2 Гбайта и кассетным НМЛ с лентой шириной 8 мм (емкость 2 Гбайта) превысит 28 тыс. ф.ст.

## Рабочая станция R260
## Фирма Acorn

*Технические характеристики*

Фирма-изготовитель: Acorn.

Наименование: R260.

Процессор: ARM3 (тактовая частота 30 МГц).

ОЗУ: 8 Мбайт с расширением до 16 Мбайт.

Порты: один параллельный, один последовательный, звуковое гнездо, видеопорт, интерфейсы Ethernet и SCSI.

Накопители: 89-мм НГМД емкостью 720 Кбайт, НМД емкостью 100 Мбайт.

Предыдущая система фирмы Acorn, R140, работавшая под управлением ОС UNIX, была оснащена ОЗУ объемом 4 Мбайта, и базируясь на процессоре ARM2 с тактовой частотой 8 МГц, не обладала достаточной мощностью для того, чтобы быть по-настоящему полезной рабочей станцией.

Новая рабочая станция R260 базируется на процессоре ARM3 с внутренней кэш-памятью, работающем с тактовой частотой 30 МГц. Это, а также использование 12-мегагерцевой шины, привело к резкому улучшению характеристик рабочей станции R260 по сравнению с предыдущей моделью. Фирма Acorn сообщает, что быстродействие R260 составляет от 10 до 12 млн. команд/сек.

На передней панели R260, корпус которой идентичен корпусу предыдущей модели R140, находятся один дисковод фирмы Citizen для 89-мм ГМД, светодиод индикации включения сетевого питания и светодиод индикации работы НМД.

Размещающийся внутри корпуса блок питания мощностью 100 Вт обеспечивает питание для видеомонитора.

На объединительной плате имеется пять гнезд для вертикально устанавливаемых плат, гнездо платы объединения модулей, на которой можно смонтировать еще четыре платы, три гнезда для плат расширения памяти и гнездо платы процессора.

В гнезда для памяти устанавливаются платы, в которых использованы 4-Мбит 80-нсек ИС ЗУПВ, на каждой плате монтируется свой блок управления памятью. Стандартные рабочие станции R260 поставляются с ОЗУ объемом 8 Мбайт, расширяемым до 16 Мбайт.

Внутри корпуса рабочей станции R260 устанавливается 89-мм НМД фирмы NEC емкостью 100 Мбайт. Место для установки дополнительных внутренних дисковых накопителей не предусмотрено.

Стандартно на плате объединения модулей монтируются платы интерфейсов Ethernet и SCSI. Плата SCSI оборудована внутренним соединителем и внешним 50-контактным разъемом.

На задней панели R260 имеются следующие порты: неподключенный соединитель Econet, 9-штырьковый последовательный порт типа D, 25-штырьковый параллельный разъем Centronix D, 3,5-мм звуковое гнездо, 9-штырьковый видеопорт типа D и три миниатюрных видеосоединителя для принудительной синхронизации с внешними источниками.

Клавиатура подключается к разъему на передней панели. Порт для манипулятора типа "мышь" (в качестве стандартного поставляется трехклавишный механический манипулятор, разработанный фирмой Logitech) находится на задней панели клавиатуры. Рядом с этим портом расположена клавиша перезапуска.

На плате видеоадаптера используется та же микросхема, что и в R140, однако добавлена дополнительная схема синхронизации, позволяющая реализовать режимы VGA и SuperVGA. Аппаратные средства отображения текстовой и графической информации поддерживают различные типы мониторов, включая 16-цветный мультичастотный режим с разрешением 640x480 элементов изображения и монохромный режим с разрешением 1152x900 элементов изображения. Использование более быстрой шины позволяет также поддерживать 16— и 256-цветный режим VGA с разрешением 640x480 и 16-цветный режим SuperVGA с разрешением 800x600 элементов изображения.

Следует отметить, что в 1991 г. фирма Acorn рассчитывает оснастить свои рабочие станции модели R260 акселератором операций с плавающей запятой собственной разработки.

Опытный образец R260 был оснащен многозадачной ОС RiscOs и ОС Risc iX 1.2, перенесенной на данную систему версией UNIX, происходящей от 4.3 BSD UNIX (Berkeley UNIX) и соответствующей стандарту X/Open XPG3. Серийные системы будут поставляться с ОС Risc iX на жестком диске.

Графические средства обеспечиваются X11r4 (сер-

вер, клиенты и пакеты разработчика), что позволяет создавать непрямоугольные окна. Используя пакет разработчика и администратор окон Motif 1.0, можно создавать изящные пиктограммы и трехмерные клавиши. Графический интерфейс пользователя осуществляется с помощью X.Desktop 2.0, а соединение с сетью — с помощью NFS 4.0 и TCP/IP.

XWindows позволяет иметь для одного физического экрана 4 виртуальных — системная консоль в символьном режиме, буфер изображения, используемый X, и еще два символьных терминала. X-сервер приспособлен под аппаратные средства R260 — он может переключать буфер изображения между четырьмя отдельными дисплеями (в дополнение к символьным экранам). Каждый из них представляется в виде отдельного X-дисплея с разрешением, соответствующим разрешению подключенного видеомонитора. Указание разрешения и переключение между дисплеями производятся программно.

Поставляемые средства разработки программ включают все необходимые библиотеки и файлы, требующиеся для написания программ для использования X11 и Motif. Основные средства программирования включают GNU Emacs 18.54 и транслятор Си, который не является, как это часто бывает на UNIX-машинах, версией транслятора Portable C compiler, а представляет собой транслятор Norcroft Ansi C, используемый для разработки ОС RiscOs. С серийными системами будут также поставляться ФОРТРАН 77 и Iso-Паскаль, причем оба используют тот же генератор объектного кода и те же оптимизации, что и транслятор Си.

Будучи совместимой с системой R140 на уровне двоичных кодов, рабочая станция R260 унаследовала все программное обеспечение предыдущей модели, включающее Informix, Uniplex, Q-Office, UNIRAS и P-GKS.

Сильной стороной машин фирмы Acorn являются их сетевые возможности. Как уже отмечалось, рабочая станция R260 оборудована встроенным портом Ethernet, предусмотрена также возможность поддержки интерфейса Econet. Работа в среде сетевой файловой системы NFS, обращение к сетевой распределенной базе данных Yellow Page, использование утилит Berkeley TCP/IP и XWindows-соединения от удаленных клиентов осуществляются без каких-либо сбоев и затруднений.

Цена описанной базовой системы с ОЗУ объемом 8 Мбайт — около 5000 ф.ст.

## Рабочая станция SystemPro
## Фирма Compaq

*Технические характеристики*

Фирма-изготовитель: Compaq.

Наименование: SystemPro.

Процессор: 2 процессора Intel 80386 (тактовая частота 33 МГц) (арифметические сопроцессоры Intel 80387 или Weitek) или 2 процессора Intel 80486 (тактовая частота 25 Мгц).

ОЗУ: 8 Мбайт с расширением до 160 Мбайт.

Порты: один параллельный Centronix, один 9-штырьковый последовательный RS-232, один порт для манипулятора типа "мышь".

Накопители: 89-мм НГМД емкостью 720 Кбайт/1,44 Мбайта, 133-мм НГМД емкостью 360 Кбайт/1,2 Мбайта, НМЛ емкостью 520 Мбайт, два НМД емкостью по 420 Мбайт.

Фирма Compaq — один из ведущих производителей ПЭВМ — выпустила на рынок мощную рабочую станцию SystemPro. Внешне эта система выглядит весьма внушительно — при высоте свыше 610 мм и ширине более 300 мм она весит около 25,5 кг, однако расположенный в верхней части корпуса скошенный отсек, в котором размещаются накопители, придает системе некоторую элегантность.

Источник питания мощностью 300 Вт, которым оборудована SystemPro, экранирован в соответствии со стандартом Федеральной комиссии связи США. Его мощности вполне хватает даже при работе системы с многочисленными последовательными терминальными портами.

Наиболее интересной особенностью машины является ее процессорный блок. Он состоит из двух процессоров Intel 80386 (33 МГц) или двух процессоров Intel 80486 (25 МГц). В перспективе планируется выпуск модели SystemPro на базе процессоров Intel 80486, работающих с тактовой частотой 33 МГц. Описываемая в настоящем обзоре система оборудована двумя процессорами 80386, один из которых работает с арифметическим сопроцессором Intel 80387, а другой — с арифметическим сопроцессором фирмы Weitek.

Благодаря тому, что каждый из процессоров установлен на отдельной плате, устанавливаемой в одно из четырех специально модифицированных расширительных гнезд, объединительная плата смонтирована с относительно невысоким уровнем плотности. Почти вся сборка произведена методом поверхностного монтажа.

На платах процессоров отсутствуют перемычки и переключатели типа DIP — вся конфигурация выполняется автоматически системными программными средствами. Первый процессор, плата которого устанавливается в гнездо 1, обеспечивает обработку прерываний, исходящих от основной системной платы. Именно этот процессор может быть перенастроен на более низкую тактовую частоту, чтобы обеспечить совместимость с более ранними DOS-системами. Что же касается ОС UNIX, оба процессора являются идентичными, и только небольшая модификация ядра обеспечивает обработку всех прерываний процессором 1.

Все платы процессоров 80386 оборудованы гнездами для установки арифметического сопроцессора Intel 80387 или Weitek. ОС SCO UNIX автоматически распознает эти дополнительные процессоры и использует их для своих математических библиотек. Процессор фирмы Weitek является более мощным (и более дорогим), чем процессор Intel 80387, но он требует уста-

| Фирма-изготовитель | Название модели | Тип микропроцессора | Быстродействие, млн.оп. в 1 с | Тактовая частота, МГц | Емкость ОЗУ, Мбайт min/max |
|---|---|---|---|---|---|
| Data General | AViiON | 88000 | 17 | 16,67 или 20 | 4/28 |
| Digital Equipment | DECstation 2100 | R2000 | 10 | 12,5 | 8/24 |
| | DECstation 3100 | R2000 | 14 | 16,67 | 8/24 |
| Hewlett-Packard | Model 340/360/370 | 68030 | 4/5/8 | 16,7/25/33 | 4/16 |
| | Model 834CH | HP-PA (фирменный с архитектурой RISC) | 14,8 | 15 | 8/48 |
| Отделение Apollo Systems фирмы Hewlett-Packard | Series 2500 Personal Workstation | 68030 | 4 | 25 | 4/16 |
| | Series 3500 Personal Workstation | 68030 | 5 | 25 | 4/32 |
| Отделение Apollo Systems фирмы Hewlett-Packard | Series 4500 Personal Super Workstation | 68030 | 8 | 33 | 8/32 |
| Integraph | Interpro 125 | Clipper | 4 | 25 | 6/12 |
| MIPS Computer | RS 2030 | R2000 | 12 | 16,67 | 8/16 |
| Opus | Personal Mainframe/8000 | 88000 | 17 | 20 | 4/20 |
| Silicon Graphics | Personal Iris | R3000 | 16 | 20 | 8/- |
| Solbourne Computer | Series 4/500 | SPARC | от 9,5 до 17 | 16,67 | 16/96 |
| Sony Microsystems | News 1700 series | 68030 | 4,3 | 25 | 4/32 |
| Sun Microsystems | SPARCstation 1 | SPARC | 12,5 | 20 | 4/16 |
| | Sun 3/80 | 68030 | 3 | 20 | 4/16 |
| | Sun 386i/150 и 250 | 80386 | 3/5 | 20/25 | 4/16 |
| Tektronix | XD 88/10 | 88000 | 17 | 20 | 8/32 |

новки дополнительной математической библиотеки. Поскольку ОС UNIX не отличает один процессор от другого, она предполагает, что если один процессор оборудован арифметическим сопроцессором, то и второй тоже. В SystemPro на базе процессоров 80486 никаких осложнений не возникает, поскольку арифметический сопроцессор интегрирован в одном кристалле с основным.

| Разрешающая способность экрана, эл. изображения | | Тип микропроцессора для операций с плавающей запятой | Емкость ЗУ на жестких дисках, Мбайт | Цена, тыс.долл. |
|---|---|---|---|---|
| монохромный | цветной | | | |
| 1280x1024 | 1280x1024x256 | внутренний | 160 или 320 | 7,45 |
| 1024x864 | 1024x864x356 | R2010 | 2x105 | 7,95 |
| 1024x864 | 1024x864x356 | R2010 | 2x105 | 11,9 |
| 1024x768 | 1024x768x16<br>1280x1024x256 | 68882 | от 81 до 571 | 8,5 |
| 1280x1024 | 1280x1024x256 | специализированный | от 156 до 300 | 19,375 |
| 1024x800<br>1280x1024 | | 68882 | от 100 до 200 | 3,99 |
| 1024x800<br>1280x1024 | 1024x800<br>1280x1024<br>16 или 256 цветов | 68882 | от 155 до 697 | 8,495 |
| 1024x800<br>1280x1024 | 1024x800<br>1280x1024<br>16 или 256 цветов | 68882 | от 155 до 697 | 19,99 |
| | 1184x884x32 | внутреиний | 156 | 16 |
| 1280x1024 | | R2010 | 172 | 17 |
| 640x480x256<br>1664x1200 | 800x600x256<br>1280x1024x256 | внутренний | 40 | 9,995 |
| | 1024x768x256<br>1280x1024x256 | R3010 | от 150 до 344 | 13,5 |
| 1152x900 | 1152x900x256 | Weitek 1164165 | 260 | 17,1 |
| 816x1024 | 1280x1024x16<br>или 256 | 68882 | 286 | 11,495 |
| 1152x900 | 1152x900x256 | Weitek 1164165 | от 104 до 1100 | 8,995 |
| 1152x900 | 1152x900x256 | 68882 | от 104 до 1100 | 5,995 |
| 1024x768 | 1024x768x256 | 80387 | от 91 до 327 | 8,99 |
| | 1280x1024x256 | внутренний | от 156 до 3000 | 16 |

Фирма Compaq сообщает, что в зависимости от конфигурации SystemPro обладает быстродействием от 8 до 40 млн. команд/сек, причем потенциал второго процессора реализуется на 80-90%.

ОЗУ машины строится на базе 80-нс 1-Мбитных ИС динамических ЗУПВ с внутренним механизмом адресации, позволяющим удерживать страницу данных объемом 2 Кбайта в течение всей продолжительности

доступа к главной памяти (режим "enhanced page" — усовершенствованная страница). В настоящее время фирма Compaq предлагает рабочие станции SystemPro, оборудованные ОЗУ объемом от 8 до 160 Мбайт (система, рассматриваемая в настоящем обзоре, снабжена ОЗУ объемом 12 Мбайт), хотя при использовании 4-Мбитных динамических ЗУПВ можно довести объем внутренней памяти до 256 Мбайт.

Центром ОЗУ является контроллер кэш 82385, к которому оба процессора обращаются с запросом о доступе к памяти. Контроллер сначала пытается обслужить этот запрос путем поиска в 64-килобитной ассоциативной кэш-памяти, использующей 25-нс статические ЗУПВ и оперирующей как с данными, так и с командами. При "непопадании" в среднем используется 6 тактов, однако фирма Compaq оценивает эффективность кэш-памяти в 97%.

На объединительной плате смонтирован набор графических микросхем, соответствующих стандарту VGA. В этот набор входит микросхема выполнения бит-блочных операций, предназначенная для высокоскоростного копирования содержания графической памяти, что обеспечивает сверхбыструю прокрутку и восстановление изображения, а также ускоряет выполнение других графических операций. Однако пользователи, желающие работать с полиэкранной системой XWindows под управлением ОС SCO UNIX, могут счесть отсутствие режима SuperVGA серьезным упущением.

Фирма Compaq стандартно комплектует рабочие станции SystemPro 35-см VGA-видеомонитором с шагом между точками 0,31мм и максимальным разрешением 640x480 элементов изображения, поддерживающим до 256 цветов.

Интересной особенностью SystemPro является конфигурация жесткого диска. В рассматриваемой системе 840 Мбайт емкости жесткого диска организованы в единый накопитель. Фактически НМД состоит из двух 420-Мбайтных НМД фирмы Western Digital, характеризующихся временем доступа 17 мсек, которые объединены на программно-аппаратном уровне. Кроме того, рассматриваемая система оборудована НМЛ емкостью 520 Мбайт.

Главным достижением здесь можно считать блок управления НМД — IDA (Intelligent Disk Array — интеллектуальная дисковая матрица). Он не только объединяет все жесткие диски, но и определяет возможность использовать любой возникающий в процессе работы нескольких накопителей параллелизм. Блок IDA может управлять четырьмя парами накопителей (два дисковода подключаются к одному и тому же IDA-интерфейсу), так что если каждый накопитель обеспечивает скорость выборки 2 Мбайта/сек, то скорость поступления данных из IDA на системную шину составит 8 Мбайт/сек. Помимо нескольких заказных микросхем, блок IDA содержит процессор 80186 и локальное ЗУ. Процессор позволяет IDA функционировать в качестве "хозяина" шины EISA, предоставляя ему особые привилегии при обращении к этой шине.

SystemPro в минимальной комплектации поставляется с одной парой 89-мм НМД, работающих как единый накопитель со скоростью передачи данных 3 Мбайта/сек. Установка каждой дополнительной пары накопителей не только повышает возможности хранения информации, но и удваивает эффективность НМД.

Для достижения параллелизма каждый файл на диске "расщеплен" (на уровне секторов) по всему числу имеющихся накопителей. При этом существует серьезная опасность того, что отказ на одном из накопителей может привести к ошибке в каждом втором, третьем или четвертом секторе каждого файла системы. Поэтому для повышения сохранности данных пришлось пойти на увеличение длины блоков. Благодаря применению системы парных дисководов возможно использование трех вариантов оперативного дублирования информации. Стандартно используется метод, в котором второй накопитель в каждой паре "зеркально" отражает информацию первого накопителя. В менее критичных применениях для дублирования данных (с соответствующим сжатием информации) используется лишь один из четырех накопителей. Если же требуется высочайшая надежность, то применяется двойное резервирование, т.е. оснащение системы двумя контроллерами и двумя наборами накопителей (в этом случае дублируется и контроллер).

Работа системы резервирования проверялась на практике. Ее эксплуатация в качестве файл-сервера в локальной сети фирмы Novell показала, что отсоединение одного из накопителей не оказывало никакого влияния на работу сети. Хотя, по сути дела, "отсутствовала" треть данных, это зафиксировал лишь блок IDA. Поэтому, когда после выключения системы было имитировано включение нового накопителя (на самом деле было восстановлено нарушенное соединение), сеть продолжала работать по-прежнему, в то время как блок IDA немедленно начал восстанавливать "утраченный" массив информации, причем вся работа выполнялась в фоновом режиме.

Рабочая станция SystemPro обладает солидными возможностями расширения — она оборудована 11 гнездами для дополнительных модулей. Из них семь гнезд поддерживают платы с архитектурой EISA, в остальные четыре могут устанавливаться стандартные модули PC/AT. В эти же гнезда могут устанавливаться патентованные платы памяти и процессоров фирмы Compaq (известные как Flex/MP).

Хотя фирма Compaq и заявляет, что вопрос использования какой-либо конкретной операционной системы является личным делом пользователя, очевидно, что рабочая станция SystemPro спроектирована для использования ОС UNIX. Данная система (SystemPro) испытывалась с последней мультипроцессорной версией ОС UNIX (System V Release 3.2) фирмы SCO, обычно обозначаемой как SCO MPX, причем для каждого из процессоров требуется своя копия SCO MPX. SCO MPX идентична стандартной ОС SCO UNIX V/386 за исключением небольших модификаций ядра

для параллельной обработки задачи двумя процессорами. Все программное обеспечение, написанное для ОС SCO UNIX V/386, будет выполняться на SystemPro без внесения в него каких-либо изменений и автоматически использовать оба процессора. В настоящее время на рынке имеется около 3000 прикладных программ, рассчитанных на применение ОС SCO UNIX. Хотя количество имеющихся графических приложений и невелико, с помощью SCO X desktop можно выполнять примерно 200 родовых X-приложений.

Прикладные программы, выполняемые под управлением ОС DOS, никоим образом не могут использовать второй процессор — природа системного программного обеспечения такова, что это возможно только для многозадачных операционных систем. В настоящее время Windows 3.0 также не может использовать оба процессора, хотя одна независимая компания в США разрабатывает версию, обладающую такими возможностями.

Стоимость SystemPro в минимальной конфигурации составляет около 12 тыс.ф.ст., а стоимость системы, описанной в настоящем обзоре, превышает 30 тыс.ф.ст.

## Рабочая станция HP Vectra 486 PC
## Фирма Hewlett-Packard

*Технические характеристики*

Фирма-изготовитель: Hewlett-Packard.

Наименование: HP Vectra 486 PC.

Процессор: 32-разрядный процессор Intel i486 (частота 25 МГц).

ОЗУ: стандартно 2 Мбайта с расширением до 64 Мбайт.

Порты: один параллельный, два последовательных, порты для подключения клавиатуры и манипулятора типа "мышь".

Накопители: поддерживается до 4-х НГМД (включая 133-мм НГМД емкостью 360 Кбайт и 1,2 Мбайта, 89-мм НГМД емкостью 1,44 Мбайта и НМЛ емкостью 120 Мбайт) и до 2-х НМД емкостью 152-670 Мбайт.

Фирма Hewlett-Packard хорошо известна в области производства средств вычислительной техники благодаря высокому качеству своих изделий — за что бы компания ни бралась, как проработка деталей, так и качество изготовления всегда соответствовуют самым высоким стандартам. Еще одним подтверждением репутации Hewlett-Packard является ее новая рабочая станция HP Vectra 486, появившаяся в конце 1989 г.

Системный блок HP Vectra 486, весом около 27 кг, выполнен в виде напольной "башни" высотой 600 мм, шириной 210 мм и глубиной 500 мм, которая покоится на опоре шириной 360 мм и глубиной 500 мм. В левом верхнем углу передней панели размещается легко доступный выключатель сетевого питания.

Внутри системного блока имеется 11 гнезд для установки модулей расширения, и, хотя часть этих гнезд занята такими обязательными платами, как платы видеоконтроллера VGA, контроллера манипулятора типа "мышь" и клавиатуры, плата памяти и объединительная плата процессора, остальные гнезда обеспечивают большие возможности расширения системы, а программа конфигурации EISA позволяет полностью определить устанавливаемые в них платы.

На объединительной плате HWPFC09 смонтирован процессор i486 (тактовая частота 25 МГц), работающий с внутренней кэш-памятью. Эта плата, а также плата расширения памяти HWP0800 рассматриваются системой как встроенные. На HP Vectra 486 поддерживаются два способа расширения памяти: использование 2— и 8-мегабайтных расширительных комплектов позволяет довести объем ОЗУ до 32 Мбайт, а использование 16-мегабайтных комплектов — до 64 Мбайт. Третьей встроенной платой является многофункциональная плата HWP1C20, обеспечивающая выполнение операций ввода-вывода для клавиатуры и манипулятора типа "мышь".

Остальные восемь гнезд соответствуют стандарту EISA и могут использоваться для 8-, 16— и 32-разрядного оборудования, соответствующего стандарту EISA или ISA. Шесть гнезд конфигурированы для работы с платами-"хозяевами" шины, а оставшиеся два гнезда обычно используются для установки плат видеоконтроллера.

Управление стандартным 36-см 16-цветным VGA-видеомонитором с разрешением 640x480 элементов изображения осуществляется платой VGA-адаптера HWP0040, которая поддерживает также режим IBM MDA, CGA, EGA и Hercules. Для решения прикладных задач, предъявляющих повышенные требования к графике (например, САПР или настольные издательские системы), рабочая станция HP Vectra 486 оснащается 16— или 20-дюймовым цветным видеомонитором высокого разрешения (до 1280x1024 элементов изображения) с шарнирным основанием. Такой видеомонитор управляется усовершенствованным графическим контроллером HP A1086 Intelligent Graphic Controller, имеющим собственное 512-килобайтное динамическое ЗУПВ.

Дополнительные возможности ввода-вывода обеспечиваются установкой коммуникационной платы HWP1C10, на которой находятся второй 9-штырьковый последовательный порт RS-232 и 25-контактный параллельный порт.

Одно из EISA-гнезд занято платой ESDI-контроллера жесткого диска HP D1664A (20 Мбит/сек), который может управлять двумя НМД емкостью до 670 Мбайт каждый.

И 102-клавишная клавиатура, и двухклавишный манипулятор типа "мышь" также изготовлены фирмой Hewlett-Packard, и работать с ними не только удобно, но и приятно.

Рабочая станция HP Vectra 486 может работать как под управлением ОС SCO UNIX System V 3.2, так и под управлением версии ОС Microsoft MS-DOS 4.01 для ПЭВМ HP Vectra 486. Эта версия включает ряд

дополнительных утилит (Personal Application Manager, Multiple Character Set Utilities, программа Disk Cache). В комплект поставки входит гибкий диск, содержащий служебную программу конфигурации EISA — EASY CONFIG. Конфигурация является двухуровневой: на верхнем уровне пользователь с помощью меню может определить, какие платы установлены в гнезда EISA, на нижнем уровне пользователь может добавлять и (или) удалять платы и загружать память любой необходимой конфигурационной информацией или программным обеспечением, например, драйверами устройств. С помощью этой программы можно также производить загрузку альтернативной операционной системы и получать доступ к некоторым утилитам системы.

По мнению экспертов журнала "PC Magazine", HP Vectra 486 на сегодня является самой мощной ПЭВМ. При использовании в качестве автономной машины, она особенно пригодна для работы с базирующимися на Windows приложениями с использованием видеомонитора высокого разрешения с большим экраном, в частности, для удовлетворения требованиям пользователей мощных настольных издательских систем и САПР.

*Н.Голуб, В.Файнберг*

**По материалам:**
Electronic Business, 23/VII, № 14, 1990.
OI Informatique, 10/X, № 1086, 1989.
EDN, 9/XI, № 23, 1989.
G.Swarbrick "IBM POWERstation 320", PC World, April 1990.
N.Walker "Sony 3860 port of call", PC Magazine, February, 1990.
S.Borgoine "Compaq SystemPro", PC World, August, 1990.
S.Littlewood "Acorn R260", PC World, August, 1990.
M.Banks "The peak of PC power", PC Magazine, April, 1990.

Появился новый цветной сканер с разрешающей способностью 400 точек на дюйм. Британская компания Colorgraph сообщает, что уже имеющийся в продаже сканер SC 7500 совместим со всеми основными графическими пакетами.

Фирма получила права на распространение в Великобритании этого устройства, которое производится тайваньской компанией Shinko. Сканер стоит от 5400 до 6500 фунтов в зависимости от интерфейса.

Размер сканера 64,5 x 55,8 x 16,3 см. Цветоделение достигается за счет использования специального цветочувствительного элемента и люминисцентного осветителя, генерирующего требуемый спектр освещения. Сканер поддерживает несколько режимов работы, в том числе режим распознавания 256 и 16 цветов, градаций серого цвета и линий. Устройство может работать с разрешающей способностью 25, 150, 200, 300 и 400 точек на дюйм. Поддерживаемые интерфейсы — GPIB (IEEE) и SCSI.

*Newsbytes News Network, 3 Jan, 1991*

Фирмы Bell и Howell Quintar разработали Quintar Page Printer Controller System (QPPCS) — новый высокоскоростной , контроллер лазерного принтера, выполненный на современном RISC-процессоре семейства AMD Am29000.

По сообщению компании, это устройство позволит более чем в 30 раз увеличить производительность монохромных и цветных принтеров. Контроллер может эмулировать большинство популярных моделей — PCL 4 и 5, IBM Proprinter XL-24, Epson FX-80, Toshiba P321 и HPGL, а также язык описания страниц PostScript. Первые образцы контроллеров будут выпущены в начале 1991 г.

*Newsbytes News Network, 2 Jan, 1991*

Фирма IBM впервые решила вынести производство компьютеров PS/2 за пределы Соединенных Штатов Америки, подписав договор с бразильской фирмой SID.

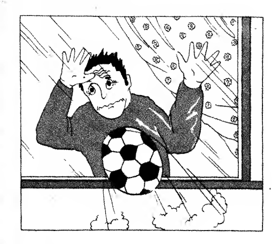
В последнее время Бразилия предприняла большие усилия с тем, чтобы сделать свой рынок — один из крупнейших в Латинской Америке — открытым для зарубежных предпринимателей. Это стало возможным после снятия в 1990 г. большинства установленных ранее ограничений.

По имеющимся данным, IBM, помимо 1 млн. долл. капитала, предоставит новому предприятию всю необходимую технологию. Для Бразилии это первая с 1984 г. сделка, при которой передается не только капитал, но и современная технология.

*Newsbytes News Network, 11 Jan, 1991*

По данным опроса 100000 пользователей, компьютеры с процессором 80286 составляют сейчас около 40% всех используемых ПК. По-прежнему используются около 2,5 млн. выпущенных ранее машин на процессорах 8088/8086. Исследование также показало, что в планах закупок техники ведущими американскими компаниями компьютеры на процессоре 80486 составляют всего 0,1%.

*Computer Reseller News, 14 Jan, 1991*



# Средства организации многооконного интерфейса

Наиболее распространенными языками программирования при разработке программного обеспечения (ПО) для персональных компьютеров (ПК) класса IBM PC являются Бейсик, Паскаль, Модула-2 и Си. Неотъемлемым компонентом каждой программной системы является ее пользовательский интерфейс. При разработке ПО на этих языках возможны два подхода. Один заключается в применении аппарата ESC-последовательностей и возможностей драйвера ANSI.SYS. Другой — в применении разнообразных инструментальных средств, позволяющих организовать гораздо более эффективный интерфейс пользователя при затрате значительно меньших усилий на его создание. Рассмотрение именно этих средств и является нашей целью.

### Пакет Basic Windowing Toolbox (B-Window) Фирма Image Computer Systems

Инструментальный пакет Basic Windowing Toolbox (B-Window) представляет собой набор функций, которые позволяют реализовать многооконный интерфейс при программировании на Бейсике. Как обычно, окно может иметь статус открытого или закрытого. При закрытии окна восстанавливается предыдущее изображение на экране. Пакет B-Window написан на Ассемблере с целью реализации высокой скорости исполнения. Система поддерживает программы на Бейсике в сочетании с применением интерпретатора или компилятора Microsoft QuickBasic 4.0. Компилятор TurboBasic фирмы Borland не поддерживает работу с пакетом, однако в будущем этот недостаток предполагается устранить. B-Window работает как с монохромными видеоадаптерами (MDA, Hercules), так и с цветными (CGA, EGA, VGA).

При определении окна задается один из шести имеющихся типов рамок. Для каждого объявленного окна формируется файл поддержки, который помещается в некоторую область памяти, называемую буфером окна. Помимо этого для функционирования окна необходим доступ к видеопамяти на том участке экрана, который перекрывает данное окно. Расположение окна на экране легко изменяется по желанию программиста.

Цена системы 20 долл. Дистрибутивная версия B-Window обеспечивает полную поддержку интерпретаторов Basic, Basica, GWBasic и им подобных, а также поддерживает работу с библиотекой QBV4.QLB компилятора QuickBasic 4.0. Ведутся работы по включению системы B-Window в интегрированное окружение этого транслятора. Зарегистрированным пользователям система будет поставляться с реализованной возможностью создания автономных EXE-файлов и генерации исходных кодов Ассемблера.

### Пакет Turbo Professional Фирма Borland International

Turbo Professional, пакет интегрированного типа, — это наиболее распространенная система для расширения возможностей стандартных библиотек Turbo Pascal, аналогичная системе Repertoire для Модулы 2. Пакет дает пользователю возможность производить действия, не предусмотренные в системе Turbo Pascal. Библиотека Turbo Professional вклю-

чает в себя ряд процедур, позволяющих:

— организовать интерфейс пользователя (т.е. создавать на экране дисплея окна меню и обрабатывать информацию в них);

— производить расширенную обработку символьных переменных;

— дополнить стандартную математическую библиотеку (например, увеличить точность вычислений и т.п.);

— работать с расширенной памятью компьютера, создавать резидентные программы и многое, многое другое.

Мы рассмотрим более подробно лишь первый пункт, поскольку он имеет непосредственное отношение к теме нашего обзора. Процедуры, включающие функции по организации интерфейса пользователя, можно разделить на следующие основные группы:

— процедуры создания окон и работы с ними;

— процедуры, позволяющие строить системы меню;

— процедуры поддержки виртуальных экранов.

По возможностям работы с окнами система Turbo Professional значительно превосходит пакет Power Tools Plus, поскольку входящие в нее процедуры позволяют осуществлять множество дополнительных сервисных операций, например, перемещать курсор и изменять его форму; прокручивать окна вверх и вниз; считывать информацию из окон и записывать ее в видеопамять с высокой скоростью; сохранять и восстанавливать окна; перемещать и изменять их параметры; включать и выключать режимы 43 линий для видеоадаптера EGA и 50 для VGA без очистки экрана и т.п. Кроме того, основным отличием данного пакета является поддержка виртуальных экранов. Для этого предусматривается возможность рассмотрения любого отрезка оперативной памяти как видеобуфера, содержимое которого может быть выведено на экран. Обычно виртуальные экраны располагают в добавочных видеостанциях или помещают буфер в свободную оперативную память. Максимальное количество одновременно существующих виртуальных экранов может быть равным 10. Это позволяет формировать изображение "за кадром", а затем мгновенно пересылать его на экран, что создает видимость молниеносной реакции программы на действия пользователя.

По возможностям работы с меню пакет Turbo Professinal мало чем отличается от аналогичных систем и поддерживает 5 основных типов меню: горизонтальные, вертикальные, меню Lotus-стиля, иерархические и типа сетки.

В заключение отметим, что пакет Turbo Professional является одной из наиболее мощных систем подобного класса.

### Пакет Power Tools Plus
### Фирма Blaise Computing

Среди прочих инструментальных средств по разработке интерфейса на языке Pascal, наибольшего внимания заслуживает система Power Tools Plus, предназначенная для работы с Turbo Pascal различных версий. Пакет является интегрированным — он обеспечивает не только поддержку программирования интерфейса пользователя, но и ряд других возможностей, например, поддержку печатающего устройства, управление памятью, обработку прерываний и т.п. Однако организация интерфейса пользователя (ИП) занимает в нем центральное место.

Пакет Power Tools Plus обеспечивает следующие возможности для программирования ИП: поддержку экрана, управление окнами, организацию работы с меню. Процедуры, входящие в библиотеку Power Tools Plus, позволяют выполнять следующие основные функции:

— создание окон с различными типами рамок и заголовков;

— обеспечение наличия собственного курсора в каждом окне;

— наложение окон друг на друга;

— организация такого количества окон, которое ограничено лишь объемом свободной оперативной памяти.

Таким образом, процедуры работы с окнами позволяют построить простейший пользовательский интерфейс без необходимости программирования на низком уровне.

В составе пакета имеется возможность создания различных типов меню: горизонтального, вертикального, типа системы Lotus и сетчатого. Поскольку имеются процедуры, позволяющие создавать меню с учетом конкретных задач пользователя (определение назначений клавиш реакции и т.п.), это ставит данный пакет в один ряд со средствами, ориентированными специально на работу с многооконным пользовательским интерфейсом. Но так как данная система является интегрированной (т.е. содержит и дополнительные процедуры, не связанные с созданием системы пользовательского интерфейса), она, естественно, не обладает такими мощными возможностями и столь гибким интерфейсом, как специализированные пакеты. Однако применение Power Tools Plus для создания многооконного интерфейса в небольших программах представляется целесообразным, так как на его разработку много времени не потребуется, а программа пользователя в результате будет выглядеть профессионально. Необходимо также отметить, что пакет довольно прост в освоении.

Наиболее распространенными системами программирования для языка Модула-2 являются TopSpeed Modula-2 фирмы J&P International, известная также под названием JPI Modula-2, и Logitech Modula-2 фирмы Logitech. Существенным доводом в пользу версии Модулы-2 фирмы Logitech является наличие на рынке программного обеспечения дополнительных инструментальных средств, значительно расширяющих ее возможности. К таким средствам относятся система

Repertoire фирмы PMI и система Window Machine (WM) фирмы Amber System Inc, которые позволяют дополнить программу множеством сервисных функций — от организации интерфейса пользователя до работы с базами данных.

### Пакет Repertoire
### Фирма PMI, Charles Bradford and Cole Brecheen

С момента своего появления в 1985 г. система Repertoire была реализована в пяти версиях и на сегодняшний день превратилась в наиболее совершенную и надежную инструментальную систему, широко применяемую во всем мире при программировании на Модуле-2 и специально ориентированную на организацию интерфейса пользователя. Она обеспечивает независимость разрабатываемой программы как на уровне исходного, так и объектного кодов; исходный код работает без изменений с любым компилятором Модулы-2, объектный — с любым Microsoft-совместимым языком. Система включает в себя необычно мощный экранный конструктор, в состав которого входят анализатор естественного языка, текстовый редактор, а также интенсивная поддержка манипуляции строками.

Система Repertoire обеспечивает выполнение трех основных задач:

— работу с расширенной памятью (Expanded Memory Specification); при этом программы пользователя получают возможность поддерживать работу с памятью, расширенной до 8 Мбайт. Наличие таких модулей позволяет совместно с использованием компилятора Modula-2 Logitech создавать системы, аналогичные Lotus 1-2-3, которые могут работать с большим числом данных, выходящих за мегабайтную границу оперативной памяти;

— создание баз данных и работу с ними, причем данные имеют формат широко распространенной СУБД dBASE III;

— организацию работы в графическом режиме (Graphix). При этом предусмотрена возможность использования объектно-ориентированной системы Meta Window (фирма MetaGraphix), которая обладает очень широким набором функций для построения графических изображений, поддерживает около 20 типов графических адаптеров и обеспечивает работу с графическими окнами.

### Пакет Window Machine (WM)
### Фирма Amber Systems

Система WM предназначена для создания многооконного интерфейса пользователя (Virtual Screen Interface) в текстовых режимах работы дисплея. Она проста в использовании и занимает всего лишь 100 Кбайт памяти на жестком диске. Система позволяет работать одновременно с 255 окнами и осуществляет все типичные для многооконных систем операции.

### Пакет C Tools Plus
### Фирма Blaise Computing

Одним из самых распространенных инструментальных средств при программировании на языке Си является пакет C Tools Plus. Последняя версия системы — C Tools Plus 6.0 — появилась на рынке программного обеспечения в конце 1989 г. Дистрибутив системы занимает 5 дискет емкостью по 360 Кбайт. В ее состав входят файлы включения, исходные тексты программ, библиотеки функций для различных моделей памяти и файлы, обеспечивающие демонстрацию возможностей системы и получение помощи.

Версия 6.0 C Tools Plus предназначена для работы с системами программирования на языке Си фирмы Microsoft версий 5.0 и 5.1, а также с QuickC версий 1.00, 1.01 и 2.00. Кроме того, специально для системы QuickC созданы две базы данных Ct6help1.hlp и Ct6help2.hlp для обеспечения возможности получения пользователем контекстуально-зависимой помощи. Главным файлом определения для создания окон является bwindow.h, в котором содержатся объявления трех основных структур, описывающих параметры программируемых окон. Преимуществом данной системы по сравнению с другими подобными средствами является мощная поддержка функций редактирования информации в окнах. Для создания системы помощи пользователю в составе C Tools Plus имеется ряд функций и структур. При этом предусмотрена возможность выделения цветом наиболее важной информации, а также возможность создания баз данных, содержащих информацию, предназначенную для вывода на экран в режиме помощи. С применением C Tools Plus возможно создание самых разнообразных меню. Все пользовательские меню поддерживают работу с манипулятором "мышь". Кроме перечисленных возможностей имеются также средства, позволяющие производить различные манипуляции с курсором, перемещать окна по экрану, изменять их размеры и т.п. Система поддерживает видеоадаптеры Hercules, CGA, EGA, MCGA, VGA и целый ряд других, менее распространенных.

### Пакет C_SCAPE
### Фирма Oakland Group

Позволяет пользователю работать с элементами данных экрана, меню, заголовками и т.п. Все операции могут производиться с экраном больше стандартного размера (т.е. 25 строк по 80 позиций), так как система обеспечивает вертикальную и горизонтальную прокрутку экрана. Она, по существу, является редактором интерфейса пользователя, позволяющим заносить информацию в свободное поле экрана дисплея. В дополнение к этому система обеспечивает такие специфические функции, как определение поля, его редактирование и рисование линий. Конечным результатом работы по созданию пользовательского интерфейса являются программы на языке Си, которые впоследствии могут быть оттранслированы и помещены в биб-

лиотеку для дальнейшего использования. Описание и создание окон осуществляется путем выполнения программы LNF. EXE. После этого можно приступать к редактированию интерфейса выбором варианта из предлагаемого меню.

Наиболее важной операцией является работа с полем. Для создания поля необходимо сначала маркировать его, а затем по подсказке, появляющейся на экране, выбрать тип заносимых данных или же создать новый. В режиме рисования можно создать на экране линии различного вида (одиночные, двойные или символьные). Поле редактируемого экрана превышает стандартный размер и содержит 100 колонок по 100 строк каждая. Операции с блоками позволяют перемещать, дублировать и удалять маркированные части текста. Одновременно с этим можно измененять цветовые атрибуты всего блока или только его маркированной части и увеличивать маркированную часть до размеров всего поля. Загрузка, сохранение файлов, а также перевод созданного разработчиком интерфейса в программу, написанную на языке Си, осуществляются выбором операции DISK из главного меню.

### Пакет C Power Window Фирма Entelekon

Пакет предназначен для организации дружественного пользовательского интерфейса на основе работы с окнами. При этом имеется возможность осуществлять следующие операции: самостоятельно организовывать меню или использовать один из предлагаемых видов; перемещать созданные окна на экране; уменьшать или увеличивать размеры окон; считывать данные из файла, находящегося на диске, в окно и наоборот из окна заносить информацию в файл; заносить информацию из окна в память или из памяти в окно; очищать окно целиком или его часть от данных; изменять атрибуты цвета всего окна или его части; выделять подсветкой часть данных в окне; организовывать наложение и пересечение окон на экране без потери информации; устанавливать приоритет окна. Система содержит исходные тексты программ, написанных на языках Си и Ассемблер, оттранслировав которые, пользователь может получить библиотеку программ для различных моделей памяти. Основной для C Power Window является концепция "виртуального экрана". Она заключается в следующем: все множество операций, производимых с окнами, осуществляется сначала в рамках "виртуального экрана" системы, который затем методом "перекачки" передается на экран монитора, что обеспечивает практически мгновенную смену информации на экране. Организация окон основывается на соглашениях, принятых в языке Си, и вся работа, осуществляемая пользователем с окнами, проводится со структурами типа WINDOW и FILEWW, определенными в файле window.h. Дополнительные структуры типа BORDERCH и BORDERSTR по умолчанию включены в структуру WINDOW и дают информацию о видах рамок окон. Фундаментальной концепцией системы организации окон в C POWER WINDOW является положение о статусе окна, которое заключается в том, что окно может быть "существующим", "готовым" или "активным".

### Пакет Window for C Фирма Vermont Creative Software

Инструментальный пакет Window for C включает в себя библиотеки, позволяющие пользователю организовать систему различных меню и окон для общения с ПЭВМ. Этот пакет является единственным из всех рассматриваемых в данной статье, который обеспечивает возможность работы с операционными оболочками TopView и Microsoft Windows. При этом программа пользователя, обращающаяся к библиотекам системы, может выполняться в режиме фоновой задачи. Система рассчитана на применение компилятора языка Си фирмы Microsoft версии 2.00 и выше. Она предоставляет пользователю следующие возможности:

— организация такого числа окон, которое ограничено только ресурсами ПЭВМ;

— использование файлов неограниченной длины, записанных в коде ASCII, для просмотра в окне (при этом осуществляется автоматическая горизонтальная и вертикальная прокрутка окна);

— организация перекрытия окон в зависимости от приоритета окна (ключа в структуре WINDOW);

— организация пользовательского меню, которое может иметь длину, превышающую размеры окна;

— использование функций работы со строками, которые позволяют размещать или изменять информацию на экране;

— создание буфера экрана, который облегчает управление и выдачу на экран файлов, содержащихся в памяти. Эти файлы могут быть любого размера, и данные могут динамически пересылаться на экран;

— запоминание информации о параметрах окна в структуре WINDOW, которая занимает около 50 байт (сами исполняемые модули для работы с окнами имеют малые размеры, что является преимуществом данной системы);

— определение цвета и вида границ окна пользователем (система также позволяет высвечивать часть текста в окне заданными атрибутами);

— занесение информации в окно по заданным ключам с переносом слов и автоматической прокруткой полного экрана;

— копирование информации, содержащейся в окне, в файл или на печатающее устройство в заданном формате;

— использование функций, позволяющих работать с клавиатурой;

— считывание и сохранение содержащихся в окне символов и их атрибутов для дальнейшего использования;

Отметим, что система позволяет работать с двумя графическими адаптерами: EGA и Monochrome Display

Adapter, а также содержит библиотеку для работы с графикой.

## Пакет Vitamin C
## Фирма Creative Programming Consultants

Является наиболее мощной системой, предназначенной для создания многооконного интерфейса. Ее библиотеки включают почти 200 функций. Она содержит файлы с исходными текстами модулей, библиотеки для различных моделей памяти, демонстрационные примеры, а также несколько сервисных программ. Как и все системы подобного класса, Vitamin C включает в себя несколько групп функций, позволяющих работать с:

— окнами (изменение размеров окна, его перемещение, изменение атрибутов части или всего окна, наложение окон друг на друга, выделение яркостью или цветом части информации в окне, очистку части или всего экрана, а также занесение необходимой информации в поле верхней и нижней границы окна; осуществление горизонтальной или вертикальной прокрутки окон);

— файлами (считывание информации из файла и занесение ее в окно с заданными или определяемыми по умолчанию атрибутами, занесение информации из окон в файл);

— курсором (установка размеров курсора, установка позиции курсора в окне по абсолютным координатам; относительное перемещение курсора в окне, прокрутка содержимого окна вверх или вниз на заданное количество строк и вправо или влево на заданное количество позиций, считывание позиции курсора в окне, занесение и считывание информации, начиная с текущей позиции курсора);

— клавиатурой (очистка буфера клавиатуры, выдача кода нажатой клавиши, помещение считанного символа в текущую позицию курсора).

Сервисные функции позволяют организовать считывание строки символов и занесение ее в окно с заданными или установленными по умолчанию атрибутами, определить день недели, число, месяц, год, а также занести эту информацию в окно и т.д.).

Таким образом, система Vitamin C предоставляет разработчику ПО довольно разнообразные возможности в решении задач по организации интерфейса пользователя.

Отметим, что пакеты Vitamin C и Window for C имеют библиотеки функций работы с окнами для всех моделей памяти, используемых в языке программирования Си, в пакет же C Power Window такие библиотеки не включены. Принципиальное различие между системами обусловлено разными подходами к организации многооконного интерфейса пользователя. Система C_SCAPE, например, позволяет увидеть на экране то, что будет получено при использовании данного фрагмента в программе, а системы Window for C, C Power Window, C Tools Plus и Vitamin C для инициализации окон предоставляют готовые структуры, включающие одни и те же параметры — координаты окон, вид границ, атрибуты поля и границ, вид курсора, возможности горизонтальной и вертикальной прокрутки и т.п. Как правило, такая структура занимает около 50 байт памяти и ее изначально заданные установки можно изменять в процессе работы программы. Библиотеки систем существенно отличаются друг от друга. Лишь система Window for C содержит графические функции, необходимые для построения на экране диаграмм простейшего вида. Наиболее характерной особенностью систем C_SCAPE, C Tools Plus и C Power Window является наличие в них функций, обеспечивающих построение готовых меню сразу нескольких видов; система Vitamin C предоставляет возможность извлечения и занесения в окна дополнительной информации, как, например, дни недели, числа, месяцы и т.п. По остальным своим возможностям системы разнятся незначительно. С точки зрения простоты создания прикладных программ на основе многооконного интерфейса, все системы требуют от пользователя определенной квалификации и опыта программирования на языке Си. Они значительно отличаются друг от друга по величине создаваемого загрузочного кода. Размер кода минимален у систем Window for C и C Power Window (30 Кбайт) для программ, использующих лишь основные операции по манипулированию окнами (инициализация, занесение информации в окно из файла и занесение информации в окно с клавиатуры, выделение информации в окне, перемещение и изменение размеров окна, осуществление перекрытия окон, их удаление). Пакеты Vitamin C и C_SCAPE за счет своей универсальности при тех же условиях дают код, приближающийся по размерам к 100 Кбайтам. Промежуточной по данным характеристикам системой является C Tools Plus, у которой загрузочный код составляет около 60-70 Кбайт.

*С.Ансилевский*
*А.Смородинский*

**По материалам:**

Software Exchange News. PC Report, 1989, June.
Pascal, Ada & Modula-2, 1988, №6.
New Products: Vitamin C Enhanced. The C User's Journal, 1989, №6.
New Products: C TOOLS PLUS /5.0. The C User's Journal, 1989, №6.
Neil Freeman. Vitamin C: A Comprehensive Screen Handling Library. The C User's Journal, 1988, №4.

*В прошлом номере мы говорили об общих вопросах и проблемах, встающих перед программистом при разработке оконного интерфейса, о правилах работы с ассемблером. Сегодня мы даем описание библиотеки для создания оконного интерфейса.*

# Создание библиотеки оконного интерфейса

**Функция get_choice()**

Функция выводит в окно вертикального меню курсор (если это необходимо) и ждет дальнейших действий пользователя. Функция позволяет перемещать курсор в пределах окна и возвращает значение выбора, который может быть произведен либо при помощи курсора, либо при помощи нажатия "горячей" клавиши, либо нажатием дополнительной функциональной клавиши. В последнем случае код нажатой клавиши возвращается в младших 9 битах (с 0 до 8), а значение текущего положения курсора в окне размещается с 9 по 15 бит. В составе функции get_choice() также применены процедуры встроенного ассемблера для оптимизации повторяющихся вызовов функций.

**Функция edit_string()**

Позволяет редактировать в окне строку неограниченной длины. При выходе строки за границы окна соответствующая часть строки сдвигается и становится невидимой, а на границе строки устанавливается маркер.

Редактор строки работает довольно непривычным образом, позаимствованным в аналогичном редакторе Turbo C. Это проявляется в том случае, если длина строки превышает размеры окна, и редактируется средняя часть строки. Идея заключается в том, чтобы редактируемый участок строки всегда был в поле зрения пользователя. Поэтому при удалении символов, в зависимости от взаимного расположения окна и строки относительно друг друга, осуществляются сдвижки как правой, так и левой части строки. Редактор работает всегда в режиме вставки символов. Отредактированная строка возвращается в том же буфере, в котором она находилась до редакции. Так как функции edit_string() в качестве параметра передается размер буфера строки (полный), то нет опасения, что возвращаемая строка превысит размер буфера. В случае, если строка превышает максимальную длину, символьные клавиши перестают реагировать. Можно отказаться от дальнейшего редактирования, нажав клавишу Escape, строка при этом остается неизменной. Нажатие на клавишу PgUp в процессе редактирования приводит к восстановлению в окне первоначального варианта строки.

Следующая серия функций выполнена целиком на встроенном ассемблере Turbo C. Здесь имеются некоторые особенности. Во всех функциях встроенного ассемблера нет операторов возврата return. В том случае, если функция должна возвращать какое-либо значение, оно перед возвратом просто помещается в те регистры, которые определены соглашениями Turbo C для соответствующих типов данных. Поэтому для таких функций компилятор Turbo C++ выдает предупреждения о том, что функция должна возвращать значение. Этих предупреждений можно было бы избежать, воспользовавшись регистровыми псевдопеременными и ставя на выходе из функции оператор return (например, return(_AX);), но наличие дополнительного оператора, на наш взгляд, не ускорит ход выполнения функции, кроме того, возникнут проблемы с описанием четырехбайтовых переменных (например, типа long). Таким образом, предлагается просто проигнорировать такие предупреждения.

**Функция break_off()**

Отключает стандартную программу обработки прерывания 1Bh (Ctrl-Break), устанавливая новый вектор прерывания, указывающий на инструкцию IRET, которая находится в теле самой функции break_off().

Таким образом, при возникновении прерывания Ctrl-Break управление передается в тело функции break_off() на инструкцию IRET. Она возвращает управление по адресу, с которого было вызвано прерывание.

Функция возвращает длинное целое, содержащее старый вектор прерывания Ctrl-Break. Разумеется, перед выходом из прикладной программы необходимо восстановить старый вектор прерывания Ctrl-Break, иначе в дальнейшем поведение компьютера может стать непредсказуемым. Следует иметь в виду, что функцию break_off() и подобные ей нельзя использовать с оверлейными структурами (по тем же соображениям).

### Функция break_on()

Служит дополнением к функции break_off(). Она восстанавливает старый вектор прерывания 1Bh (Ctrl-Break), переданный ей в качестве входного параметра, либо устанавливает новый.

### Функция get_video_mode()

Возвращает значение текущего режима работы дисплея, которое считывается из соответствующего байта нулевого сегмента.

Рассмотрим теперь функции, широко использующие информацию, которую BIOS (базовая система ввода-вывода) заносит в низкоуровневую память. В результате отпадает необходимость в частом обращении к прерываниям BIOS для получения такой информации, и, следовательно, можно организовать работу программы более рациональным способом.

### Функция toggle_intensity_blinking()

Переключает бит мигания/интенсивности видеоадаптеров EGA/VGA, обращаясь к соответствующему прерыванию BIOS. Таким образом, появляется возможность использовать в стандартной палитре цветов для текстовых режимов темно-серый цвет для фоновых атрибутов символа. Функция не работает с другими типами видеоадаптеров, отличными от EGA или VGA.

### Функция getkey()

Осуществляет вызов прерывания 16h, 00h BIOS, стандартная программа обработки которого ожидает нажатия какой-либо клавиши и возвращает ее ASCII и скан-коды. Далее getkey() преобразовывает полученные коды способом, часто используемым для подобных целей. Если ASCII-код клавиши не равен нулю, то функция просто возвращает ASCII-код. В противном случае функция возвращает скан-код нажатой клавиши, прибавив к нему 256 (установив нулевой бит старшего байта в 1). Таким образом, достигается решение сразу двух задач. Во-первых, те клавиши, которые имеют на клавиатуре своих двойников (например, белые и серые стрелки, клавиши Home, End и т.д.), получают одинаковый возвращаемый код и, таким образом, отпадает необходимость в обрабатывающей программе учитывать различия между данными клавишами. (При этом некоторые комбинации клавиш становятся недоступными, но они используются довольно редко). Во-вторых, четко известно, что код любой нажатой клавиши будет лежать в диапазоне от 0 до 512 (в младших 9 битах), и, следовательно, старшие 7 бит того же слова можно каким-либо образом использовать (что и осуществляется, например, в функции get_choice()).

### Функция get_cursor_size()

Определяет форму аппаратного (мигающего) текстового курсора и возвращает величину целого типа, содержащую номера конечной и начальной скан-линий курсора.

### Функция set_cursor_size()

Является дополнением к функции get_cursor_size() и позволяет задать форму аппаратного текстового курсора. Если в качестве входного параметра функции set_cursor_size() передать величину 2000h, то мигающий курсор станет невидимым, оставаясь при этом на той же позиции экрана. Некоторые программисты советуют для получения невидимого курсора убирать его за рамки экрана — на 25 (26) строку, опасаясь, что в противном случае на некоторых компьютерах старых образцов могут возникнуть определенные побочные эффекты. Однако описанный способ получения невидимого курсора является обычным для большинства программных продуктов.

В библиотеке стандартных функций Turbo C++ имеется аналогичная функция _setcursortype(), позволяющая устанавливать три различных формы курсора: высокий, низкий, либо невидимый. однако эта функция не имеет дополняющей ее функции, позволяющей прочитать текущую форму курсора. Кроме того, функции _setcursortype() нет в реализации Turbo C 2.0.

### Функция set_cursor_position()

Совершенно идентична стандартной библиотечной функции Turbo C gotoxy(). Функция set_cursor_position() была введена в состав библиотеки оконного интерфейса только из-за того, что по каким-то причинам функция gotoxy() не отслеживает переключений страниц дисплея и всегда перемещает курсор только на нулевой странице.

Функция set_cursor_position() при переключении страниц работает должным образом. При передаче функции координат курсора, выходящих за рамки, установленные для текущего текстового режима (80x25), поведение функции не определено.

Следующие функции — get_cursor_position_size() и set_cursor_position_size() — являются взаимодополняющими и разработаны специально для использования в составе функции edit_string() и ей подобных.

Функция get_cursor_position_size() возвращает длинное целое, содержащее все текущие параметры аппа-

ратного курсора, т.е. его форму и позицию на экране (причем для данных двух функций, в отличие от функции set_cursor_position(), позиция курсора отсчитывается от нулевой строки и нулевого столбца), а функция set_cursor_position_size() при передаче ей той же величины в качестве входного параметра установит курсор в точно такое же состояние, как и то, в котором он находился перед вызовом комплемеитарной функции. Таким образом, появляется удобная возможность сохранять и затем восстанавливать состояние аппаратного курсора на экране в случае необходимости произвести с ним какие-либо промежуточные манипуляции.

Для следующего набора функций, осуществляющих вывод информации на экран посредством прямых обращений к видеопамяти, необходимо пояснить некоторые принципы организации видеопамяти для текстовых режимов работы дисплея.

Для цветного текстового режима с размерами экранного поля 80 столбцов по горизонтали и 25 строк по вертикали (режимы 2 и 3) видеопамять начинается с сегментного адреса 0B800h, а для аналогичного монохромного режима (режим 7) — с сегментного адреса 0B000h. Максимальное количество переключаемых текстовых страниц дисплея равно 8 и они имеют номера с 0 по 7. Каждая страница занимает объем памяти 4000 байт, а каждая последующая страница смещена относительно предыдущей на 4 Кбайта. Таким образом, для цветиого текстового режима сегментные адреса страниц будут следующими: 0B800h для нулевой страницы, 0B900h — для первой, 0BA00h — для второй и т.д. Небольшие участки памяти, остающиеся между концом предыдущей страницы и началом следующей (разница между 4 Кбайтами и 4000 байт), являются резервными и никак не используются.

Видеопамять отдельной экранной страницы организована следующим образом. Все символьное поле экрана представлено в виде одной длинной строки, в которой каждая последующая строка экрана начинается сразу после предыдущей и ничем от нее не отделена. Каждая символьная клетка экрана описывается в видеопамяти двумя байтами. В первом байте размещается ASCII-код представляемого на экране символа, во втором размещаются атрибуты цвета этого символа, и далее коды символов и их атрибутов последовательно чередуются. Таким образом, каждая полная экранная строка будет занимать в видеопамяти 160 байт (для режима 80x25) и каждый символ некоторого столбца последующей строки будет смещен относительно символа того же столбца предыдущей строки тоже на 160 байт.

Байт атрибутов цвета символа содержит информацию о цвете самого символа (в младшем полубайте) и информацию о цвете фона текущей символьной клетки (в старшем полубайте). Самый старший — 7 бит этого байта — обычно отвечает за мигание символа на экране. Для монохромного дисплея можно получить подчеркнутые символы. Для этого атрибут цвета символа (младший полубайт) должен быть равен 1h (в цветном режиме это соответствует цвету BLUE).

### Функция make_hbar()

Строит на обозначенном месте экрана маркер горизонтального курсора, заменяя атрибуты того участка экрана, который попадает под выстраиваемый курсор, на атрибуты, содержащиеся в переданной функции строке. Перед выводом курсора на экран функция сохраняет старые атрибуты экрана в строке-буфере, указатель на которую также передается функции в качестве входного параметра.

Тем, что функции make_hbar() передается сразу целая строка новых атрибутов курсора, а не отдельное значение атрибута, единое для всего курсора, достигаются сразу две цели. Во-первых, можно сделать курсор "разноцветиым", т.е. "закрасить" различные участки курсора различными атрибутами. Во-вторых, строку-буфер, в которую были помещены старые атрибуты экрана, можно будет в дальнейшем использовать как строку-источник для восстановления атрибутов экрана при удалении курсора. При этом строкой-буфером станет уже первоначальная строка атрибутов курсора, в которую вернутся те же самые атрибуты, что и были из иее прочитаны ранее. Таким образом, функция make_hbar() становится универсальной, т.е. ее можно использовать как для построения горизонтального курсора на экране, так и для его удаления. В последнем случае следует лишь поменять местами указатели на строку-источник и строку-приемник, передаваемые функции в качестве входных параметров. Необходимо отметить, что при передаче функции координат строки (или ширины курсора), выходящих за рамки экрана, в лучшем случае вы увидите курсор или его продолжение на следующих строках, а в худшем — последствия будут непредсказуемыми.

### Функция clear_nchars()

Очищает в заданном месте экрана строку указанной длииы, заполняя символьные байты видеопамяти нулями, которые на экране выглядят как пробелы.

### Функция put_string()

Выводит на экран указанную строку, иачиная с заданной позиции, пересылая символы непосредственно в видеопамять.

Две следующие функции — update_left() и update_right() — предназначены для работы в составе функции edit_string() и самостоятельно не используются. Они обновляют на экране соответственно левую или правую части редактируемой строки от текущего положения курсора и устанавливают маркеры-индикаторы, если строка выходит за рамки окна. Маркеры устанавливаются вне границ строки.

### Функция make_window()

Строит на экране окно с тенью, отбрасываемой таким образом, как если бы свет падал со стороны левой

верхней части экрана. На экран выводится текст из строки, содержащей полную построчную развертку окна вместе с рамками, но без атрибутов символов. Можно получать на экране символы с двумя различными значениями атрибутов — атрибутами, общими для окна, и атрибутами "горячих" символов.

Кажущееся на первый взгляд нерациональным размещение в исходной строке полного текста окна — вместе с рамками и пустыми промежутками — на самом деле оправдано, так как эта же строка будет в дальнейшем использована в качестве буфера для хранения измененного текста окна (после вывода в окно некоторой информации). С другой стороны, такая организация способа построения окон удобна лишь при разработке интерфейсов с не слишком большим количеством поддерживаемых ими окон, так как начиная с определенного уровня выигрыш в простоте сменяется проигрышем в объеме из-за необходимости хранить большую массу дополнительной ненужной информации (рамки и пустые промежутки).

Перед выдачей окна на экран функция make_window() пересылает информацию с участка экрана, перекрываемого окном (с тенью), в буфер. (Текст выводится вместе с атрибутами.)

Для рационального построения программы в теле функции make_window() организуются два счетчика, один из которых следит за номерами "горячих" символов, а другой — за правой границей строки окна. Цикл выдачи символов на экран инициализируется каждый раз по тому счетчику, значение которого ближе к текущей позиции последнего выведенного символа.

Для функции make_window() существует простой способ вычисления номеров "горячих" символов в строке текста окна. В любом текстовом редакторе нужно просто выстроить эту строку в полную ее длину (т.е. так, как она и будет храниться в памяти) и подвести курсор к желаемому символу. Счетчик горизонтальной позиции курсора покажет вам нужный номер.

### Функция get_window_text()

Является дополнением к функции make_window(). Она считывает текст без атрибутов из окна на экране и пересылает его в указанный буфер. В качестве буфера может использоваться исходная строка текста окна, в которую в этом случае вернется уже каким-то образом измененная информация.

### Функция restore_text()

Аналогична стандартной библиотечной функции Turbo C puttext() с той разницей, что функция puttext() почему-то не отслеживает переключений страниц на экране и загружаемый модуль, получаемый при компиляции программы с включенной в нее функцией puttext(), получается почти на целый килобайт длиннее, чем при использовании функции restore_text(). Функция restore_text() между тем не осуществляет проверку, лежат ли значения переданных ей координат в установленном для них диапазоне.

Функция restore_text() восстанавливает информацию на участке экрана, расположенном под окном, используя для этого текст из строки-буфера, заполняемой при построении окна функцией make_window(). Однако, с одинаковым успехом можно использовать функцию restore_text() и для самостоятельного построения окон, заменив ею библиотечную функцию Turbo C puttext().

### Функция insert_char()

Осуществляет вставку символа в ту позицию строки, указатель на которую был передан функции в качестве входного параметра. Остальные символы строки сдвигаются при этом вправо.

### Функция delete_char()

Удаляет символ из той позиции строки, указатель на которую был ей передан в качестве входного параметра. Оставшиеся символы строки сдвигаются при этом влево.

### Функция string_copy()

Аналогична стандартной библиотечной функции Си strcpy(). Различие заключается в том, что функция string_copy() возвращает длину скопированной строки, во многих случаях это удобно, так как исчезает необходимость в вызове функции Си strlen() для определения длины строки. Функция string_copy() используется в составе функции edit_string().

В заключение необходимо добавить, что данный вариант оконного интерфейса разрабатывался не с целью создания некой универсальной библиотеки, а с целью получения простого и эффективного средства для построения систем пользовательских меню в составе небольших по объему программ. Представленная в качестве примера в статье программа, скомпилированная с моделью памяти Small, вырабатывает загружаемый модуль объемом около 21,5 Кбайта, из которых на долю непосредственно пользовательского интерфейса приходится примерно 7 Кбайт исполняемого кода плюс память, используемая под окна.

Что касается вопросов дальнейшего совершенствования приведенного выше варианта библиотеки, удобно было бы переработать функции построения окон таким образом, чтобы можно было описывать каждое окно стандартным способом, используя для этого определенным образом сконфигурированные структуры, указатели на которые будут передаваться в качестве входных параметров соответствующим функциям, т.е. так, как это организовано в большинстве профессиональных пакетов. Создание же дополнительных функций, осуществляющих, например, работу с системами горизонтальных меню, не вызовет у пользователя значительных затруднений, так как в библиотеке присутствуют все необходимые для этих целей компоненты.

```
/* Файл UTILIT3.C */
/* Автор А.Синев, Copyright (C) 1990,1991 */
/* Turbo C 2.0, Turbo C++ 1.0 */

#pragma inline

#include <alloc.h>
#include <string.h>

#include "makeprt.h"

/********** Получить значение выбора **********/
/********** в окне вертикального меню *********/
/* Функция возвращает значение выбора. Если была
   нажата специальная клавиша, возвращается также
   ее код в младших 9 битах (с 0 по 8), значение
   выбора при этом размещается с 9 по 15 биты.
   Параметры:
   first_row - первая строка меню на экране;
   last_row  - последняя строка меню на экране;
   start_col - первый столбец курсора меню на
               экране;
   bar_width - ширина курсора меню;
   curr_choice - текущее значение выбора;
   sourcebar - указатель на строку атрибутов
          курсора, длиной bar_width;
   destinbar - указатель на буфер атрибутов
          курсора, длиной bar_width;
   nn_altkeys - количество альтернативных клавиш,
             равное удвоенному количеству строк
             меню (для символов нижнего и
             верхнего регистров) плюс число
             специальных функциональных клавиш,
          отсутствующих в окне меню;
   altkeys - указатель на массив кодов
          альтернативных клавиш;
   bar_status - текущее состояние курсора меню:
     0 - курсора в окне нет, 1 - курсор есть */

int get_choice(int first_row,int last_row,
    int start_col,int bar_width,int curr_choice,
    unsigned char *sourcebar,
    unsigned char *destinbar,int nn_altkeys,
    int *altkeys,int bar_status)
{
 int row;         /* текущая строка */
 int ch;          /* код символа с клавиатуры */
 register int i;  /* счетчик */

 row = first_row + curr_choice - 1;
 if (bar_status)
   goto key_loop;

loop:
 /* построить курсор в окне меню */
 asm call near ptr makecursor
key_loop:
 /* прочитать символ с клавиатуры */
 switch(ch = getkey()) {
   /* если ENTER, то выход из цикла */
   case ENTER: goto quit;
   case UPKEY:           /* стрелка вверх */
     /* восстановить атрибуты под курсором */
     asm call near ptr restorecursor
     row--;
     break;
   case DOWNKEY:          /* стрелка вниз */
     /* восстановить атрибуты под курсором */
     asm call near ptr restorecursor
     row++;
     break;
   case PGUPKEY:          /* клавиша PgUp */
   case HOMEKEY:          /* клавиша Home */
   case LEFTKEY:          /* стрелка влево */
     asm call near ptr restorecursor
     row = first_row;
     break;
   case PGDNKEY:          /* клавиша PgDn */
   case ENDKEY:           /* клавиша End */
   case RIGHTKEY:         /* стрелка вправо */
     asm call near ptr restorecursor
     row = last_row;
     break;
   /* для остальных кодов проверять соответствие
     альтернативным клавишам */
   default:
     for (i = 0; i<((last_row-first_row + 1)<<1);
                                   i += 2) {
       if (ch == altkeys[i] ||
                 ch == altkeys[i + 1]) {
         /* если соответствие найдено,
         восстановить атрибуты под курсором */
         asm call near ptr restorecursor
       row = first_row + (i >> 1);
         /* построить курсор в новом месте */
         asm call near ptr makecursor
       goto quit;     /* переход на возврат */
       }
     }
     /* проверять соответствие кодам специальных
       клавиш, отсутствующих в окне меню */
     for (; i<nn_altkeys; i++) {
       /* если соответствие найдено, возвращается
          целая величина, в которой младшие 9 бит
          (с 0 по 8) отведены под код специальной
       клавиши, а с 9 по 15 биты размещается
       текущее значение выбора */
       if (ch == altkeys[i])
       return (((row-first_row + 1) << 9) |
                          altkeys[i]);
     }
     /* если соответствие не найдено,
       возврат в цикл чтения с клавиатуры */
     goto key_loop;
 }
 /* если выбор за рамками окна - произвести
   коррекцию */
 if(row < first_row)
   row = last_row;
 else {
   if(row > last_row)
     row = first_row;
 }
 goto loop;    /* возврат в цикл */

quit:
 /* вернуть значение выбора */
 return (row - first_row + 1);

/*---------- Процедуры встроенного ассемблера */
 /* процедура построения курсора */
 asm makecursor proc near
 make_hbar(row,start_col,bar_width,sourcebar,
                       destinbar);
 asm ret
 asm makecursor endp

 /* процедура восстановления атрибутов экрана
   под курсором */
```

```
 asm restorecursor proc near
 make_hbar(row,start_col,bar_width,destinbar,
                        sourcebar);
 asm ret
 asm restorecursor endp

} /* get_choice() */


/**** Функция редактирования строки в окне ****/
/* Функция позволяет редактировать в окне
   заданного размера строку неограниченной длины,
   сдвигая ее влево и/или вправо и помечая выход
   строки за рамки окна. Функция возвращает
   отредактированную строку в том же буфере, в
   котором она была передана.
   Параметры:
   row       - номер строки окна на экране;
   start_col - левая координата окна на экране;
   end_col   - правая координата окна на экране;
   cursorshape - форма курсора;
   buffersize - размер буфера строки, байт;
   originalstring - указатель на редактируемую
            строку;
   sourceattr - указатель на строку атрибутов
         для закрашиваемой части окна;
   destattr - указатель на буфер атрибутов окна;
   Размеры массивов sourceattr[] и destattr[]
   должны быть равны (end_col - start_col - 2).
   Функция возвращает 0, если строка не редакти-
   ровалась (была нажата клавиша ESCAPE), либо
   1, если строка была отредактирована. */

int edit_string(int row,int start_col,
   int end_col,int cursorshape,int buffersize,
   char *originalstring,unsigned char *sourceattr,
   unsigned char *destattr)
{
 int ch;                    /* код символа */
 /* начальная и конечная координаты строки на
    экране, ширина окна */
 int startcolumn,endcolumn,width;
 /* индикаторы выхода строки за рамки окна */
 int beg_status,end_status;
 /* текущая длина строки, длина остатка строки
    справа от курсора */
 int length,rest;
 char *buffer;  /* буфер для редактирования */
 int entry;      /* номер вхождения в цикл */
 /* старые параметры курсора: x и y координаты
    и форма курсора */
 unsigned long cursorparms;
 int cursorpos_w;  /* позиция курсора в окне */
 /* указатель на позицию курсора в строке */
 char *cursorpos_s;
 int ret_status;   /* возвращаемое значение */

 /* зарезервировать буфер для редактирования
    строки размера buffersize */
 if ((buffer = malloc(buffersize)) == NULL)
   return 0;

 /* сохранить старые параметры курсора */
 cursorparms = get_cursor_position_size();

 /* вычислить промежуточные значения */
 startcolumn = start_col + 1;
 endcolumn = end_col - 1;
 width = end_col - startcolumn;
 /* максимальная длина строки */
 buffersize--;
 /* скопировать строку в буфер и получить ее
    длину, задать начальное значение остатка */
 length = string_copy(buffer,originalstring);
 rest = 0;

 /* начальное положение курсора в строке - в
    конце строки, нач.знач. правого индикатора */
 cursorpos_s = buffer + length;
 end_status = 0;

 /* определить положение курсора в окне и
    значение левого индикатора */
 if (length < width) {
   cursorpos_w = startcolumn + length;
   beg_status = 0;
 } else {
   cursorpos_w = endcolumn;
   beg_status = 1;
 }
 /* вывести строку на экран */
 asm call near ptr updateleft
 /* закрасить строку атрибутами *sourceattr для
    индикации нередактированной строки */
 if (length)
   make_hbar(row,startcolumn,cursorpos_w -
            startcolumn,sourceattr,destattr);
 /* установить курсор на нужную позицию */
 set_cursor_position(cursorpos_w,row);
 /* установить форму курсора */
 set_cursor_size(cursorshape);
 entry = 0;      /* нулевое вхождение в цикл */

key_loop:
 /* прочитать код с клавиатуры */
 switch(ch = getkey()) {
  case ESCAPE:       /* нажата клавиша ESCAPE */
   ret_status = 0;
   goto quit;       /* перейти на возврат */
  case ENTER:         /* нажата клавиша ENTER */
   ret_status = 1;
   /* преобразовать отредактированную строку в
      верхний регистр и скопировать ее в
      первоначальный буфер */
   strupr(buffer);
   strcpy(originalstring,buffer);
   goto quit;       /* перейти на возврат */
  case HOMEKEY:       /* нажата клавиша Home */
   /* если нулевое вхождение, то восстановить
      атрибуты экрана */
   if (!entry) {
     if (length)
       asm call near ptr restorebar;
     entry = 1;
   }
   if (rest != length) {
     cursorpos_w = startcolumn;
     cursorpos_s = buffer;
     rest = length;
     beg_status = 0;
     end_status = (rest <= width) ? 0 : 1;
     /* обновить строку вправо */
     asm call near ptr updateright
     break;
   } else
     goto key_loop;
  case PGUPKEY:       /* нажата клавиша PgUp */
   if (!entry) {
     if (length)
```

```
      asm call near ptr restorebar;
    entry = 1;
  }
  length = rest = string_copy(buffer,
                          originalstring);
  cursorpos_w = startcolumn;
  cursorpos_s = buffer;
  beg_status = 0;
  end_status = (length <= width) ? 0 : 1;
  clear_nchars(row,startcolumn,width);
  asm call near ptr updateright
  break;
case LEFTKEY:    /* нажата стрелка влево */
  if (!entry) {
    if (length)
      asm call near ptr restorebar;
    entry = 1;
  }
  if (cursorpos_w > startcolumn) {
    cursorpos_w--;
    cursorpos_s--;
    rest++;
    break;
  } else {
    if (rest < length) {
   cursorpos_s--;
   rest++;
   beg_status = (rest == length) ? 0 : 1;
   end_status = (rest <= width) ? 0 : 1;
      asm call near ptr updateright
    }
    goto key_loop;
  }
case ENDKEY:    /* нажата клавиша END */
  if (!entry) {
    if (length)
      asm call near ptr restorebar;
    entry = 1;
  }
  if (rest) {
    cursorpos_s = buffer + length;
    end_status = 0;
    rest = 0;
    if (length < width) {
   cursorpos_w = startcolumn + length;
   beg_status = 0;
    } else {
   cursorpos_w = endcolumn;
   beg_status = 1;
    }
    /* обновить строку влево */
    asm call near ptr updateleft
    break;
  } else
    goto key_loop;
case RIGHTKEY:    /* нажата стрелка вправо */
  if (!entry) {
    if (length)
      asm call near ptr restorebar;
    entry = 1;
  }
  if (cursorpos_w < endcolumn) {
    if (rest) {
   cursorpos_w++;
   cursorpos_s++;
   rest--;
   break;
    }
    goto key_loop;
  } else {
    if (rest) {
   cursorpos_s++;
   rest--;
   beg_status = ((length-rest) <width) ? 0 : 1;
   end_status = (rest<=1) ? 0 : 1;
      asm call near ptr updateleft
    }
    goto key_loop;
  }
case DELKEY:    /* нажата клавиша DELETE */
  if (!entry) {
    if (length)
      asm call near ptr restorebar;
    entry = 1;
  }
  if (rest) {
    delete_char(cursorpos_s,rest);
    rest--;
    length--;
    if (rest < endcolumn-cursorpos_w) {
   end_status = 0;
   if (beg_status) {
     cursorpos_w++;
     beg_status = (length-rest <
           cursorpos_w-start_col) ? 0 : 1;
       asm call near ptr updateleft
     break;
   }
      asm call near ptr updateright
    } else {
   end_status = (rest <=
             end_col-cursorpos_w) ? 0 : 1;
      asm call near ptr updateright
    }
  }
  goto key_loop;
case BACKSPACE: /* нажата клавиша BACKSPACE */
  if (!entry) {
    if (length)
      asm call near ptr restorebar;
    entry = 1;
  }
  if (rest < length) {
    cursorpos_s--;
    delete_char(cursorpos_s,rest+1);
    length--;
    if (rest < end_col-cursorpos_w) {
   end_status = 0;
   if (beg_status) {
     beg_status = (length-rest <
           cursorpos_w-start_col) ? 0 : 1;
       asm call near ptr updateleft
   } else {
     cursorpos_w--;
       asm call near ptr updateright
     break;
   }
    } else {
   if (cursorpos_w > startcolumn) {
     cursorpos_w--;
     end_status = (rest <=
             end_col-cursorpos_w) ? 0 : 1;
       asm call near ptr updateright
     break;
   } else {
     if (rest == length) {
       beg_status = 0;
         asm call near ptr updateright
```

```
      }
     }
    }
   }
   goto key_loop;

  default:          /* для остальных клавиш */
   /* если нажата "незаконная" клавиша, то
     возврат в цикл */
   if (ch<32 || ch>255)
    goto key_loop;
   if (!entry) {   /* если нулевое вхождение */
    if (length)
     asm call near ptr restorebar;
    *buffer = 0;
    length = rest = beg_status = end_status = 0;
    cursorpos_s = buffer;
    cursorpos_w = startcolumn;
    entry = 1;
    /* очистить строку на экране */
    clear_nchars(row,start_col,
              end_col-start_col+1);
    set_cursor_position(startcolumn,row);
   }
   /* если буфер полон, вернуться в цикл */
   if (length >= buffersize)
    goto key_loop;
   /* вставить символ в строку */
   insert_char(cursorpos_s,rest,ch);
   length++;
   if (cursorpos_w < endcolumn) {
    end_status = (rest <
              end_col-cursorpos_w) ? 0 : 1;
    asm call near ptr updateright
    cursorpos_s++;
    cursorpos_w++;
    break;
   }
   cursorpos_s++;
   beg_status = (length-rest <
            cursorpos_w-start_col) ? 0 : 1;
   asm call near ptr updateleft
   goto key_loop;
  }
  /* установить курсор на нужную позицию */
  set_cursor_position(cursorpos_w,row);
  goto key_loop;

quit:
 free(buffer);       /* освободить буфер */
 /* восстановить параметры курсора */
 set_cursor_position_size(cursorparms);
 return ret_status;  /* возврат */

/*---------- Процедуры встроенного ассемблера */
 /* процедура восстановления атрибутов
   закрашенного бруска */
 asm restorebar proc near
 make_hbar(row,startcolumn,cursorpos_w -
      startcolumn,destattr,sourceattr);
 asm ret
 asm restorebar endp

 /* процедура обновления строки вправо */
 asm updateright proc near
 update_right(row,startcolumn,endcolumn,
        beg_status,end_status,
        cursorpos_w,cursorpos_s);
 asm ret
 asm updateright endp

 /* процедура обновления строки влево */
 asm updateleft proc near
 update_left(row,startcolumn,endcolumn,
        beg_status,end_status,
       cursorpos_w,cursorpos_s);
 asm ret
 asm updateleft endp

} /* edit_string() */

/* Конец файла UTILIT3.C */


/* Файл INLUTIL.C */
/* Автор А.Синев, Copyright (C) 1990,1991 */
/* Turbo C 2.0, Turbo C++ 1.0 */

#pragma inline

/* Для всех функций, осуществляющих прямые
  обращения к видеопамяти, отсчет координат
  интересующей текстовой клетки экрана ведется
  от единицы (т.е. левая верхняя символьная
  клетка имеет координаты 1,1); все
  соответствующие параметры функций должны
  удовлетворять этому соглашению. */

/* адрес вектора прерывания 0x1b (Ctrl-Break)
  в таблице прерываний (смещение и сегмент) */
#define BreakOffsetPtr 0x1b*4
#define BreakSegmentPtr 0x1b*4+2

/************ Отключить обработку *************/
/******** прерывания 0x1b (Ctrl-Break) ********/
/* Функция возвращает значение старого вектора
  прерывания 0x1b. Сегментный адрес содержится в
  старшем слове, смещение - в младшем */
unsigned long break_off(void)
{
 asm    xor ax,ax;
 asm    moves,ax;   /* 0 сегмент в ES */

 asm    cli; /* запретить обработку прерываний */

 /* сохранить старый вектор в DX:AX */
 /* смещение */
 asm    movax,es:BreakOffsetPtr;
 /* сегмент */
 asm    movdx,es:BreakSegmentPtr;

 /* установить новый вектор, указывающий на
   инструкцию IRET */
 /* смещение */
 asm    movword ptr es:BreakOffsetPtr,offset nobreak;
 /* сегмент */
 asm    moves:BreakSegmentPtr,cs;

 asm    sti; /* разрешить обработку прерываний */
 asm    jmp short quit; /* переход на возврат */

 /* подпрограмма обработки прерывания,
   возвращающая управление по адресу вызова */
asm nobreak:
 asm    iret;

quit:;      /* возврат из функции */
} /* break_off() */
```

```
/******* Установить программу обработки *******/
/******** прерывания 0x1b (Ctrl-Break) ********/
/* Функция устанавливает новый вектор обработки
   прерывания 0x1b (либо восстанавливает старый),
   переданный ей через параметр oldbreakvector,
   где сегментный адрес содержится в старшем слове,
   а смещение - в младшем */
void break_on(unsigned long oldbreakvector)
{
 asm    xor ax,ax;
 asm    moves,ax;    /* 0 сегмент в ES */

 asm    cli;  /* запретить обработку прерываний */

 /* записать вектор в таблицу прерываний */
 /* смещение */
 asm    movax,oldbreakvector;
 asm    moves:BreakOffsetPtr,ax;
 /* сегмент */
 asm    movax,oldbreakvector[2];
 asm    moves:BreakSegmentPtr,ax;

 asm    sti;  /* разрешить обработку прерываний */
} /* break_on() */

/******** Прочитать текущий видеорежим ********/
/* Функция возвращает значение текущего
   видеорежима в регистре AX */
int get_video_mode(void)
{
 asm    xor ax,ax;
 asm    moves,ax;       /* 0 сегмент в ES */
 /* поместить байт текущего видеорежима
    (0:449h) в AX */
 asm    moval,es:[449h];    /* AH = 0 */
} /* get_video_mode() */

/**** Переключить бит мигания/интенсивности ****/
/* Функция производит переключение в соответствии
   со значением параметра blinking_status:
   1 - включить мигание;
   0 - включить интенсивность.
   Функция работает только с видеоадаптерами
   EGA/VGA
*/
void toggle_intensity_blinking(int blinking_status)
{
 asm    push    sp;  /* сохранить значения */
 asm    push    bp;  /* изменяемых регистров */
 asm    push    si;
 asm    push    di;
 /* вызов прерывания 10h, 1003h "Toggle
    intensity/blinking" */
 asm    movax,1003h;
 asm    movbl,byte ptr blinking_status;
 asm    int  10h;

 asm    pop di;   /* восстановить значения */
 asm    pop si;   /* регистров */
 asm    pop bp;
 asm    pop sp;
} /* toggle_intensity_blinking() */

/********* Прочитать код с клавиатуры *********/
/* Функция возвращает код клавиши, полученный по
   прерыванию 16h, 00h и преобразованный
   следующим образом. Если ASCII-код клавиши не
   равен 0, то возвращается ASCII-код в младшем
   байте слова, старший же байт равен 0. Если
   ASCII-код клавиши равен 0, то в младшем байте
   возвращается скан-код этой клавиши, а старший
   байт устанавливается равным 01h. */
int getkey(void)
{
 /* вызов прерывания 16h, 00h "Keyboard Read" */
 /* После возврата AL содержит ASCII-код клавиши,
    AH - скан-код клавиши */
 asm    xor ah,ah;
 asm    int  16h;

 asm    andal,al;     /* проверить AL = 0 */
 /* если ASCII = 0, то переход на обработку скан
    кода */
 asm    jz  scancode;
 asm    xor ah,ah; /* очистить старший байт */
 asm    jmp short quit;   /* переход на возврат */
scancode:
 /* передать скан-код в младший байт */
 asm    moval,ah;
 /* установить старший байт равным 01h */
 asm    movah,01h;
quit:;                  /* возврат */
} /* getkey() */

/***** Определить форму аппаратного курсора *****/
/* Функция возвращает номера начальной и конечной
   скан-линиий аппаратного курсора:
   начальная скан-линия - в старшем байте слова;
   конечная скан-линия - в младшем байте слова. */
int get_cursor_size(void)
{
 asm    xor ax,ax;
 asm    moves,ax;     /* 0 сегмент в ES */

 /* конечная и начальная скан-линии курсора
    (0:460h) */
 asm    movax,es:[460h];
} /* get_cursor_size() */

/******* Задать форму аппаратного курсора *******/
/* Функция устанавливает форму аппаратного курсора
   в соответствии со значением переданного
   параметра cursor_shape:
   старший байт слова содержит номер начальной
   скан-линии курсора;
   младший байт слова содержит номер конечной
   скан-линии курсора. */
void set_cursor_size(int cursor_shape)
{
 /* передать параметры курсора в CX и вызвать
    прерывание 10h, 01h "Set cursor size/shape" */
 asm    movcx,cursor_shape;
 asm    movah,01h;
 asm    int  10h;
} /* set_cursor_size() */

/***** Установить курсор в заданную позицию *****/
/* Функция устанавливает аппаратный курсор в
   позицию, определяемую параметрами column и row
   (считая от единицы):
   column - горизонтальная позиция курсора;
   row - вертикальная позиция курсора; */
void set_cursor_position(int column, int row)
{
 asm    xor ax,ax;
 asm    moves,ax;     /* 0 сегмент в ES */
```

```
/* передать номер текущей страницы дисплея
   (0:462h) в BH */
asm    movbh,es:[462h];
/* номера столбца и строки в DX */
asm    movdl,byte ptr column;
asm    movdh,byte ptr row;
/* подстроить под базу 0 */
asm    sub dx,0101h;
/* прерывание 10h, 02h "Set cursor position" */
asm    movah,02h;
asm    int 10h;
} /* set_cursor_position() */

/******** Определить текущую позицию и ********/
/********* форму аппаратного курсора **********/
/* Функция возвращает длинное целое, содержащее
  следующие значения:
  0 байт - конечная скан-линия курсора;
  1 байт - начальная скан-линия курсора;
  2 байт - горизонтальная позиция курсора;
  3 байт - вертикальная позиция курсора.
*/
unsigned long get_cursor_position_size(void)
{
 asm    xor ax,ax;
 asm    moves,ax;      /* 0 сегмент в ES */

 /* номер текущей страницы (0:462h) в AL */
 asm    moval,es:[462h];
 /* умножить на 2 для адресации слов */
 asm    shl ax,1;
 /* стартовый адрес таблицы позиций курсора
   (0:450h) в BX */
 asm    movbx,450h;
 /* прибавить смещение для текущей страницы */
 asm    addbx,ax;
 /* горизонтальная и вертикальная позиции
   курсора в DX */
 asm    movdx,es:[bx];
 /* конечная и начальная скан-линии курсора в AX */
 asm    movax,es:[460h];
} /* get_cursor_position_size() */

/** Задать позицию и форму аппаратного курсора **/
/* Функция устанавливает курсор на экране в
  указанную позицию и задает его форму в
  соответствии с переданным параметром
  cursorparms, в котором:
  0 байт - конечная скан-линия курсора;
  1 байт - начальная скан-линия курсора;
  2 байт - горизонтальная позиция курсора;
  3 байт - вертикальная позиция курсора.
*/
void set_cursor_position_size(unsigned long cursorparms)
{
 asm    xor ax,ax;
 asm    moves,ax;      /* 0 сегмент в ES */

 /* номер текущей страницы дисплея (0:462h) в BH */
 asm    movbh,es:[462h];
 /* передать горизонтальную и вертикальную позиции
   курсора в DX */
 asm    movdx,cursorparms[2];
 /* вызов прерывания 10h,02h "Set cursor position" */
 asm    movah,02h;
 asm    int 10h;
 /* конечная и начальная скан-линии курсора в CX */
 asm    movcx,cursorparms;
 /* прерывание 10h,01h "Set cursor size/shape" */
 asm    movah,01h;
 asm    int 10h;
} /* set_cursor_position_size() */

/***** Построить на экране горизонтальный ******/
/******* брусок с указанными атрибутами ********/
/* Параметры:
  row - координата строки на экране;
  start_col - координата начального (левого)
        столбца на экране;
  width    - ширина бруска;
  sourcestring - указатель на строку новых
        атрибутов бруска (длиной width);
  deststring  - указатель на буфер для старых
        атрибутов бруска (длиной width).
*/
void make_hbar(int row, int start_col, int width,
        unsigned char far *sourcestring,
        unsigned char far *deststring)
{
 asm    push    ds;   /* сохранить значение DS */

 asm    xor ax,ax;
 asm    moves,ax;      /* 0 сегмент в ES */

 /* стартовый (сегментный) адрес видеопамяти для
   цветного текстового режима */
 asm    movdx,0b800h;
 /* проверить текущий видеорежим */
 asm    cmpbyte ptr es:[449h],7;
 /* если не монохромный режим, то переход на
   skip_mono, иначе задать соответствующий
   стартовый адрес видеопамяти для монохромного
   режима */
 asm    jne skip_mono;
 asm    movdx,0b000h;
skip_mono:
 /* передать сегментный адрес видеопамяти в DS */
 asm    movds,dx;

 /* вычислить смещение бруска в видеопамяти */
 /* передать вертикальную координату бруска в CX
   и подстроить под базу 0 */
 asm    movcx,row;
 asm    dec cx;
 /* длина экранной строки (в байтах) в AL */
 asm    moval,160;
 /* вычислить смещение строки от начала страницы
   в AX (умножив на 160) */
 asm    mul cl;
 /* передать горизонтальную позицию начала бруска
   в CX и подстроить под базу 0 */
 asm    movcx,start_col;
 asm    dec cx;
 /* умножить на 2 для доступа к словам */
 asm    shl cx,1;
 /* вычислить смещение начала бруска от текущей
   страницы в AX */
 asm    addax,cx;

 /* вычислить смещение начала бруска в сегменте
   видеопамяти, прибавив смещение текущей
   страницы (0:44eh) */
 asm    addax,es:[44eh];

 asm    movsi,ax; /* передать смещение в SI */

 /* загрузить far-указатель на строку-приемник */
 asm    les di,deststring;
```

```
 asm     movcx,width; /* ширина бруска в CX */
 /* временно сохранить ширину в BX */
 asm     movbx,cx;
 /* очистить direction-флаг для обработки строки в
    сторону увеличения адреса */
 asm     cld;
 /* сохранить старые атрибуты бруска в буфере */
video2memory:
 /* увеличить адрес на 1 для доступа к атрибутам
    символа */
 asm     inc si;
 asm     movsb;    /* передать атрибут в буфер */
 asm     loop     video2memory; /* возврат в цикл */

 /* передать адрес видеосегмента в ES */
 asm     moves,dx;
 asm     movdi,ax;   /* смещение бруска в DI */
 /* загрузить far-указатель на строку-источник */
 asm     lds si,sourcestring;
 asm     movcx,bx;  /* ширина бруска в CX */
 /* переслать новые атрибуты бруска в видеопамять */
memory2video:
 /* увеличить адрес на 1 для доступа к атрибутам
    символа */
 asm     inc di;
 asm     movsb;  /* переслать байт в видеопамять */
 asm     loop     memory2video; /* возврат в цикл */

 asm     pop ds; /* восстановить значение DS */
} /* make_hbar() */

/*** Очистить на экране строку указанной длины **/
/* Функция очищает на экране строку указанной
   длины, начиная с указанной позиции.
   Параметры:
   row, start_col - вертикальная и горизонтальная
      координаты первого символа строки на экране;
   nn_chars - количество удаляемых символов. */
void clear_nchars(int row, int start_col,
         int nn_chars)
{
 asm     xor ax,ax;
 asm     moves,ax;      /* 0 сегмент в ES */

 /* стартовый адрес видеопамяти для цветного
    текстового режима */
 asm     movcx,0b800h;
 /* проверить текущий видеорежим */
 asm     cmpbyte ptr es:[449h],7;
 /* если не монохромный режим, то переход на
    skip_mono, иначе задать соответствующий
    стартовый адрес видеопамяти для монохромного
    режима */
 asm     jne skip_mono;
 asm     movcx,0b000h;
skip_mono:
 /* подстроить вертикальную координату под базу 0 */
 asm     movdx,row;
 asm     dec dx;
 /* вычислить смещение строки */
 asm     moval,160;
 asm     mul dl;
 /* подстроить горизонтальную координату под
    базу 0 */
 asm     movdl,start_col;
 asm     dec dl;
 /* умножить на 2 для доступа к словам */
 asm     shl di,1;
 /* прибавить смещение столбца */
 asm     add di,ax;

 /* прибавить смещение текущей страницы в
    видеосегменте */
 asm     add di,es:[44eh];
 /* передать стартовый адрес видеосегмента в ES */
 asm     moves,cx;

 asm     xor ax,ax;    /* установить AL = 0 */
 /* установить счетчик символов */
 asm     movcx,nn_chars;
 /* очистить direction-флаг для движения в сторону
    увеличения адреса */
 asm     cld;
 /* очистить строку на экране */
clear_char:
 asm     stosb           /* вывести 0 на экран */
 /* пропустить атрибут символа */
 asm     inc di;
 asm     loop     clear_char; /* возврат в цикл */
} /* clear_nchars() */

/********** Вывести строку на экран ***********/
/* Функция выводит на экран ASCIIZ-строку (строка,
   заканчивающаяся нулем), начиная с указанной
   позиции. Параметры:
   row, start_col - вертикальная и горизонтальная
      координаты первого символа строки на экране;
   sourcestring - указатель на выводимую строку.
*/
void put_string(int row, int start_col,
        char far *sourcestring)
{
 asm     push     ds;    /* сохранить значение DS */

 asm     xor ax,ax;
 asm     moves,ax;      /* 0 сегмент в ES */

 /* стартовый адрес видеопамяти для цветного
    текстового режима */
 asm     movcx,0b800h;
 /* проверить текущий видеорежим */
 asm     cmpbyte ptr es:[449h],7;
 /* если не монохромный режим, то переход на
    skip_mono, иначе задать соответствующий
    стартовый адрес видеопамяти для монохромного
    режима */
 asm     jne skip_mono;
 asm     movcx,0b000h;
skip_mono:
 /* подстроить вертикальную координату под базу 0 */
 asm     movdx,row;
 asm     dec dx;
 /* вычислить смещение строки */
 asm     moval,160;
 asm     mul dl;
 /* подстроить горизонтальную координату под
    базу 0 */
 asm     movdl,start_col;
 asm     dec di;
 /* умножить на 2 для доступа к словам */
 asm     shl di,1;
 /* прибавить смещение столбца */
 asm     add di,ax;

 /* прибавить смещение текущей страницы в
    видеосегменте */
 asm     add di,es:[44eh];
 /* передать стартовый адрес видеосегмента в ES */
```

```
 asm    moves,cx;

 /* загрузить far-указатель на строку-источник */
 asm    lds si,sourcestring;
 /* очистить direction-флаг для движения в
    сторону увеличения адреса */
 asm    cld;
 /* выдать строку на экран */
memory2video:
 asm    lodsb;      /* загрузить символ в AL */
 asm    and al,al;          /* AL = 0 ? */
 /* если конец строки, то переход на возврат */
 asm    je  quit;
 asm    stosb;      /* вывести символ на экран */
 /* пропустить атрибут символа */
 asm    inc di;
 /* возврат в цикл */
 asm    jmp short memory2video;
quit:
 asm    pop ds; /* восстановить значение DS */
} /* put_string() */

/******** Обновить строку в окне влево *********/
/* Функция обновляет в окне на экране левую часть
   редактируемой строки от текущей позиции курсора,
   устанавливая одновременно маркеры, указывающие
   на продолжение строки за рамки окна (маркеры
   устанавливаются за границами строки).
   Параметры:
   row - вертикальная координата строки;
   startcolumn, endcolumn - левая и правая
      горизонтальные координаты границ строки
      на экране;
   beg_status, end_status - ключи состояний левого
      и правого маркеров:
      1 - установить маркер,
      0 - сбросить маркер;
   cursorpos_w - позиция курсора в окне
      (абсолютная горизонтальная координата на
       экране)
   cursorpos_s - указатель на текущую позицию
         курсора в строке.
*/
void update_left(int row, int startcolumn,
    int endcolumn, int beg_status, int end_status,
    int cursorpos_w, char far *cursorpos_s)
{
 asm    push    ds;    /* сохранить значение DS */

 asm    xor ax,ax;
 asm    moves,ax;      /* 0 сегмент в ES */

 /* стартовый адрес видеопамяти для цветного
    текстового режима */
 asm    movcx,0b800h;
 /* проверить текущий видеорежим */
 asm    cmpbyte ptr es:[449h],7;
 /* если не монохромный режим, то переход на
    skip_mono, иначе задать соответствующий
    стартовый адрес видеопамяти для монохромного
    режима */
 asm    jne skip_mono;
 asm    movcx,0b000h;
skip_mono:
 /* подстроить номер строки под базу 0 */
 asm    movdx,row;
 asm    dec dx;
 /* вычислить смещение строки на странице */
 asm    moval,160;
 asm    mul dl;

 /* прибавить смещение текущей страницы */
 asm    add ax,es:[44eh];
 /* адрес видеосегмента в ES */
 asm    moves,cx;

 /* позиция правого маркера в DI */
 asm    movdi,endcolumn;
 asm    shl di,1;       /* умножить на 2 */
 /* прибавить смещение страницы и строки */
 asm    add di,ax;
 /* очистить правый маркер на экране */
 asm    movbyte ptr es:[di],0;
 /* если end_status = 0, то переход */
 asm    cmpbyte ptr end_status,0;
 asm    je  skip_endsign;
 /* иначе установить правый маркер */
 asm    movbyte ptr es:[di],16;
skip_endsign:
 /* позиция курсора на экране в CX */
 asm    movcx,cursorpos_w;
 asm    movdi,cx;      /* скопировать в DI */
 asm    dec di;   /* подстроить под базу 0 */
 asm    shl di,1;      /* умножить на 2 */
 /* прибавить смещение страницы и строки */
 asm    add di,ax;

 /* загрузить far-указатель на позицию курсора в
    строке */
 asm    lds si,cursorpos_s;
 /* установить direction-флаг для обработки строки
    в сторону уменьшения адреса */
 asm    std;
 /* вычислить длину выводимой строки */
 asm    sub cx,word ptr startcolumn;
 asm    inc cx;
memory2video:              /* выводить строку */
 asm    movsb;      /* вывести символ на экран */
 asm    dec di;     /* пропустить атрибут */
 asm    loop    memory2video; /* возврат в цикл */

 /* очистить левый маркер */
 asm    movbyte ptr es:[di],0;
 /* если beg_status = 0, то переход */
 asm    cmpbyte ptr beg_status,0;
 asm    je  skip_begsign;
 /* иначе установить левый маркер */
 asm    movbyte ptr es:[di],17;
skip_begsign:
 asm    pop ds; /* восстановить значение DS */
} /* update_left() */

/******** Обновить строку в окне вправо ********/
/* Функция обновляет в окне на экране правую часть
   редактируемой строки от текущей позиции курсора,
   устанавливая одновременно маркеры, указывающие
   на продолжение строки за рамки окна (маркеры
   устанавливаются за границами строки).
   Параметры:
   row - вертикальная координата строки;
   startcolumn, endcolumn - левая и правая
      горизонтальные координаты границ строки
      на экране;
   beg_status, end_status - ключи состояний левого
      и правого маркеров:
      1 - установить маркер,
      0 - сбросить маркер;
   cursorpos_w - позиция курсора в окне
```

```
     (абсолютная горизонтальная координата на
     экране)
  cursorpos_s - указатель на текущую позицию
        курсора в строке.
*/
void update_right(int row, int startcolumn,
   int endcolumn, int beg_status, int end_status,
   int cursorpos_w, char far *cursorpos_s)
{
 asm    push    ds;   /* сохранить значение DS */

 asm    xor ax,ax;
 asm    moves,ax;      /* 0 сегмент в ES */

 /* стартовый адрес видеопамяти для цветного
   текстового режима */
 asm    movcx,0b800h;
 /* проверить текущий видеорежим */
 asm    cmpbyte ptr es:[449h],7;
 /* если не монохромный режим, то переход на
   skip_mono, иначе задать соответствующий
   стартовый адрес видеопамяти для монохромного
   режима */
 asm    jne skip_mono;
 asm    movcx,0b000h;
skip_mono:
 /* подстроить номер строки под базу 0 */
 asm    movdx,row;
 asm    dec dx;
 /* вычислить смещение строки на странице */
 asm    moval,160;
 asm    mul dl;

 /* прибавить смещение текущей страницы */
 asm    addax,es:[44eh];
 asm    moves,cx;  /* видеосегмент в ES */

 /* левая координата строки */
 asm    movdi,startcolumn;
 /* подстроить под позицию левого маркера */
 asm    dec di;
 asm    dec di;  /* подстроить под базу 0 */
 /* умножить на 2 для доступа к словам */
 asm    shl di,1;
 /* прибавить смещение страницы и строки */
 asm    add di,ax;
 /* очистить левый маркер */
 asm    movbyte ptr es:[di],0;
 /* если beg_status = 0, то переход */
 asm    cmpbyte ptr beg_status,0;
 asm    je  skip_begsign;
 /* иначе установить левый маркер */
 asm    movbyte ptr es:[di],17;
skip_begsign:
 /* правая граница строки */
 asm    movcx,endcolumn;
 /* позиция курсора на экране */
 asm    movdi,cursorpos_w;
 /* вычислить количество обновляемых символов */
 asm    sub cx,di;
 asm    inc cx;
 /* подстроить координату под базу 0 */
 asm    dec di;
 asm    shl di,1;       /* умножить на 2 */
 /* прибавить смещение страницы и строки */
 asm    add di,ax;
 /* временно сохранить смещение в BX */
 asm    movbx,ax;
 /* загрузить far-указатель на текущую позицию в
   строке */
 asm    lds si,cursorpos_s;
 /* очистить direction-флаг для движения в
   сторону увеличения адреса */
 asm    cld;
 /* выводить символы до конца строки */
memory2video:
 asm    lodsb;      /* загрузить символ в AL */
 asm    stosb;    /* вывести символ на экран */
 /* если конец строки, то выход из цикла */
 asm    and al,al;
 asm    jz  endofstring;
 asm    inc di;    /* пропустить атрибут */
 asm    loop    memory2video; /* возврат в цикл */
endofstring:
 /* координата правого маркера */
 asm    movdi,endcolumn;
 asm    shl di,1;
 /* прибавить смещение страницы и строки */
 asm    add di,bx;
 /* очистить правый маркер */
 asm    movbyte ptr es:[di],0;
 /* если end_status = 0, то переход */
 asm    cmpbyte ptr end_status,0;
 asm    je  skip_endsign;
 /* иначе установить правый маркер */
 asm    movbyte ptr es:[di],16;
skip_endsign:
 asm    pop ds; /* восстановить значение DS */
} /* update_right() */

/********** Построить окно на экране ***********/
/* Функция строит на экране окно с тенью, выводя
  в него текст из указанной строки.
  Параметры:
  left, top, right, bottom - левая, верхняя,
      правая и нижняя координаты окна на экране;
  sourcetext - указатель на строку текста окна
         (без атрибутов);
  buffer - указатель на буфер для сохранения
      участка экрана под окном;
      размер буфера с учетом тени:
      2*(right-left + 1 + 2)*(bottom-top + 1 + 1)
  charattr - атрибуты окна;
  shadowbackgroundattr - фоновые атрибуты тени
      (в старшем полубайте младшего байта);
  nn_hotkeys - количество символов, которые
     необходимо "закрасить" атрибутами hotkeyattr;
  hotkeys - указатель на массив целых чисел
      размера nn_hotkeys, содержащий номера тех
      символов в строке sourcetext (считая от 1),
      которые будут "закрашены" атрибутами
      hotkeyattr;
  hotkeyattr - атрибуты "горячих" символов в окне.
*/
void make_window(int left, int top, int right,
    int bottom, char far *sourcetext,
    char far *buffer, int charattr,
    int shadowbackgroundattr, int nn_hotkeys,
    int far *hotkeys, int hotkeyattr)
{
 /* промежуточные переменные */
 int key,i,j,prevch;

 asm    push    ds;   /* сохранить значение DS */

 asm    xor ax,ax;
 asm    moves,ax;      /* 0 сегмент в ES */
```

```
/* стартовый адрес видеопамяти для цветного
   текстового режима */
asm   movax,0b800h;
/* проверить текущий видеорежим */
asm   cmpbyte ptr es:[449h],7;
/* если не монохромный режим, то переход на
   skip_mono, иначе задать соответствующий
   стартовый адрес видеопамяти для монохромного
   режима */
asm   jne skip_mono;
asm   movax,0b000h;
skip_mono:
asm   movds,ax;  /* видеосегмент в DS */

/* подстроить верхнюю координату под базу 0 */
asm   movdx,top;
asm   dec dx;
/* вычислить смещение строки, умножив на 160 */
asm   moval,160;
asm   mul dl;
/* подстроить номер столбца под базу 0 */
asm   movsi,left;
asm   dec si;
asm   shl si,1;       /* умножить на 2 */
/* смещение окна на текущей странице */
asm   addsi,ax;

/* прибавить смещение страницы */
asm   addsi,es:[44eh];
/* временно сохранить смещение в AX */
asm   movax,si;

/* загрузить far-указатель на строку-приемник */
asm   les di,buffer;
/* нижняя координата в BX */
asm   movbx,bottom;
/* вычислить высоту окна */
asm   sub bx,top;
asm   inc bx;
/* временно сохранить высоту окна без тени в
   переменной top */
asm   movtop,bx;
/* высота окна с тенью в BX */
asm   inc bx;
/* вычислить ширину окна без тени в DX */
asm   movdx,right;
asm   sub dx,left;
asm   inc dx;
/* временно сохранить ширину окна без тени в
   переменной left */
asm   movleft,dx;
asm   inc dx;  /* прибавить ширину тени */
asm   inc dx;
/* сохранить смещение окна в видеосегменте в
   переменной right */
asm   movright,ax;
/* очистить direction-флаг для движения в
   сторону увеличения адреса */
asm   cld;
asm   jmp short skip_addition;  /* переход */
/* цикл передачи текста с экрана под окном
   в буфер (с атрибутами) */
video2mem_row:
/* вычислить смещение для левой границы
   следующей строки окна и передать в SI */
asm   addax,160;
asm   movsi,ax;
skip_addition:
/* ширина окна с тенью в CX */
asm   movcx,dx;
/* передавать строку в буфер по словам */
asm   rep movsw;
/* уменьшить счетчик строк окна на 1 */
asm   dec bx;
/* если счетчик не равен 0, то возврат в цикл */
asm   jne video2mem_row;

/* начальное значение счетчика строк окна */
i = 0;
/* если количество символов, которые будут
   закрашиваться атрибутами hotkeyattr, равно 0,
   то задать номер следующего "горячего" символа
   слишком большим */
asm   cmpword ptr nn_hotkeys,0;
asm   jne gethotkey;
key = 0x7fff;
asm   jmp short skiphotkey;

gethotkey:
/* начальное значение счетчика "горячих"
   символов */
j = 0;
/* прочитать номер первого "горячего" символа */
key = hotkeys[0];
asm   moval,charattr; /* атрибуты окна в AL */
/* атрибуты "горячих" символов в AH */
asm   movah,hotkeyattr;
/* временно сохранить атрибуты в charattr */
asm   movcharattr,ax;

skiphotkey:
/* передать адрес видеосегмента в ES */
asm   movax,ds;
asm   moves,ax;
/* передать смещение окна в DI */
asm   movdi,right;

/* загрузить far-указатель на строку текста
   окна */
asm   lds si,dword ptr sourcetext;
/* скопировать ширину окна без тени в переменную
   bottom */
asm   movax,left;
asm   movbottom,ax;
/* задать начальное значение номера предыдущего
   символа (0) */
asm   movword ptr prevch,0;

/* очистить direction-флаг для движения в
   сторону увеличения адреса */
asm   cld;

again:          /* цикл вывода окна на экран */
asm   movax,bottom; /* ширина окна в AX */
/* сравнить с номером следующего "горячего"
   символа */
asm   cmpax,key;
/* переход, если правая граница текущей строки
   окна располагается ближе, чем следующий
   "горячий" символ */
asm   jl  lineloop;

/* атрибуты окна в AL, атрибуты "горячих"
   символов в AH */
asm   movax,charattr;
/* вычислить длину строки текста окна до
   ближайшего "горячего" символа */
```

```
 asm    movcx,key;
 asm    sub cx,prevch;
 asm    dec cx;
 /* если два "горячих" символа расположены рядом,
    то переход на skipchar */
 asm    jz  skipchar;
/* выводить строку до "горячего" символа */
hotloop:
 asm    movsb;    /* вывести символ на экран */
 asm    stosb;       /* вывести атрибут окна */
 asm    loop    hotloop;  /* возврат в цикл */
skipchar:
 asm    movsb;  /* вывести "горячий" символ */
 /* передать атрибут "горячего" символа в AL */
 asm    moval,ah;
 asm    stosb; /* вывести атрибут */

 /* номер предыдущего символа равен номеру
    выведенного "горячего" символа */
 asm    movax,key;
 asm    movprevch,ax;
 /* увеличить счетчик "горячих" символов на 1 */
 j++;
 /* сравнить с количеством "горячих" символов;
    если все "горячие" символы уже выведены на
    экран, то переход на skipnextkey */
 asm    movax,nn_hotkeys;
 asm    cmpj,ax;
 asm    jge skipnextkey;

 /* временно сохранить ES в DX */
 asm    movdx,es;

 /* прочитать номер следующего "горячего" символа
    из массива hotkeys[] и записать его в key */
 asm    movax,j;
 asm    shl ax,1;
 asm    les bx,dword ptr hotkeys;
 asm    add bx,ax;
 asm    movax,word ptr es:[bx];
 asm    movkey,ax;

 asm    moves,dx;   /* восстановить ES */
 asm    jmp short again;  /* возврат в цикл */

skipnextkey:
 /* если все "горячие" символы уже выведены, то
    задать номер следующего горячего символа
    слишком большим */
 key = 0x7fff;
 asm    jmp short again;  /* возврат в цикл */

/* выводить строку до правой границы окна */
lineloop:
 /* AX = ширина_окна_без_тени * i - prevch */
 asm    sub ax,prevch;
 /* если дошли до правой границы окна, то переход
    на vertshadow */
 asm    je  vertshadow;

 /* передать в CX количество символов до правой
    границы окна */
 asm    movcx,ax;
 asm    movax,charattr; /* атрибуты окна в AL */
lineloop1:      /* выводить строку на экран */
 asm    movsb;   /* вывести символ */
 asm    stosb;   /* вывести атрибут */
 asm    loop    lineloop1;  /* возврат в цикл */
```

```
/* строить с правой стороны окна вертикальную
   тень шириной в 2 колонки */
vertshadow:
 if (i) {  /* начинать со второй строки окна */
  /* атрибуты тени в DL */
  asm mov    dx,shadowbackgroundattr;
  /* установить в младшем полубайте DL все
     единицы */
  asm or  dl,0fh;
  asm mov    dh,0f0h;  /* маска 0f0h в DH */
  asm inc di;      /* пропустить символ */
  /* прочитать атрибут символа с экрана */
  asm moval,es:[di];
  /* установить в старшем полубайте AL все
     единицы */
  asm or  al,dh;
  /* изменить фоновые атрибуты символа на атрибуты
     тени */
  asm and     al,dl;
  /* вывести измененные атрибуты символа на экран */
  asm stosb;
  /* повторить процедуру для следующего символа */
  asm inc di;
  asm moval,es:[di];
  asm or  al,dh;
  asm and     al,dl;
  asm stosb;
 }
 /* переслать номер последнего выведенного символа
    в prevch */
 asm    movax,bottom;
 asm    movprevch,ax;
 /* вычислить номер символа на правой границе
    следующей строки окна и переслать в bottom */
 asm    addax,left;
 asm    movbottom,ax;
 i++;         /* увеличить счетчик строк на 1 */
 /* вычислить смещение следующей строки и
    загрузить в DI */
 right += 160;
 asm    mov     di,right;
 /* если не последняя строка окна, то вернуться
    в цикл */
 if (i<top)
   goto again;

 /* строить горизонтальную тень вдоль нижней
    границы окна */
 asm    adddi,4; /* пропустить два символа */
 asm    movcx,left;  /* ширина окна в CX */
horizshadow:
 asm    inc di;   /* пропустить символ */
 /* загрузить атрибут в AL */
 asm    moval,es:[di];
 /* изменить фоновые атрибуты на атрибуты тени */
 asm    or  al,dh;
 asm    and     al,dl;
 asm    stosb; /* вывести атрибут на экран */
 asm    loop    horizshadow; /* вернуться в цикл */

 asm    pop ds; /* восстановить значение DS */
} /* make_window() */

/******** Прочитать текст из окна на **********/
/********** экране (без атрибутов) ************/
/* Функция считывает из окна с указанными
   координатами текст без атрибутов символов и
   заполняет им указанную строку.
   Параметры:
```

```
   left, top, right, bottom - левая, верхняя,
       правая и нижняя координаты окна на экране;
   desttext - указатель на строку для заполнения;
         длина строки должна быть равна:
         (right-left + 1)*(bottom-top + 1)
*/
void get_window_text(int left, int top, int right,
         int bottom, char far *desttext)
{
 asm    push    ds;   /* сохранить значение DS */

 asm    xor ax,ax;
 asm    moves,ax;     /* 0 сегмент в ES */

 /* стартовый адрес видеопамяти для цветного
   текстового режима */
 asm    movax,0b800h;
 /* проверить текущий видеорежим */
 asm    cmpbyte ptr es:[449h],7;
 /* если не монохромный режим, то переход иа
   skip_mono, иначе задать соответствующий
   стартовый адрес.видеопамяти для монохромного
   режима */
 asm    jne skip_mono;
 asm    movax,0b000h;
skip_mono:
 asm    movds,ax;   /* видеосегмент в DS */

 /* подстроить верхнюю координату под базу 0 */
 asm    movdx,top;
 asm    dec dx;
 /* вычислить смещение строки на странице, умножив
   на 160 */
 asm    moval,160;
 asm    mul dl;
 /* подстроить левую координату под базу 0 */
 asm    movsl,left;
 asm    dec si;
 /* вычислить смещение окна на странице и передать
   в SI */
 asm    shl si,1;
 asm    addsi,ax;

 /* прибавить смещение текущей страницы */
 asm    addsi,es:[44eh];
 /* временно сохранить смещение в AX */
 asm    movax,si;

 /* загрузить far-указатель на строку-приемник */
 asm    les di,desttext;

 /* вычислить высоту окна в BX */
 asm    movbx,bottom;
 asm    sub bx,top;
 asm    inc bx;
 /* вычислить ширину окна в DX */
 asm    movdx,right;
 asm    sub dx,left;
 asm    inc dx;

 /* очистить direction-флаг для движения в
   сторону увеличения адреса */
 asm    cid;
 asm    jmp short skip_addition;
/* цикл считывания строк окна */
video2mem_row:
 /* вычислить смещение следующей строки окна и
   передать в SI */
 asm    addax,160;
 asm    movsi,ax;
skip_addition:
 asm    movcx,dx;   /* ширина окна в CX */
again:    /* цикл считывания символов строки */
 asm    movsb;   /* переслать символ в буфер */
 asm    inc si;    /* пропустить атрибут */
 asm    loop    again; /* возврат в цикл */

 /* уменьшить счетчик строк на 1 */
 asm    dec bx;
 /* если счетчик не равен 0, то возврат в цикл */
 asm    jne video2mem_row;

 asm    pop ds; /* восстановить значение DS */
} /* get_window_text() */

/******** Восстановить текст на экране ********/
/* Функция заполняет окно с указанными координатами
   на экране текстом из указанной строки. Строка
   содержит коды символов, чередующиеся с их
   атрибутами. Параметры:
   left, top, right, bottom - левая, верхняя,
      правая и нижняя координаты окна на экране;
   sourcetext - указатель на строку кодов символов
         и их атрибутов.
*/
void restore_text(int left, int top, Int right,
        int bottom, char far *sourcetext)
{
 asm    push    ds;   /* сохранить значение DS */

 asm    xor ax,ax;
 asm    moves,ax;     /* 0 сегмент в ES */

 /* стартовый адрес видеопамяти для цветного
   текстового режима */
 asm    movcx,0b800h;
 /* проверить текущий видеорежим */
 asm    cmpbyte ptr es:[449h],7;
 /* если не монохромный режим, то переход на
   skip_mono, иначе задать соответствующий
   стартовый адрес видеопамяти для монохромного
   режима */
 asm    jne skip_mono;
 asm    movcx,0b000h;
skip_mono:
 /* подстроить верхнюю координату под базу 0 */
 asm.   movdx,top;
 asm    dec dx;
 /* вычислить смещение строки на странице, умножив
   на 160 */
 asm    moval,160;
 asm    mul dl;
 /* подстроить левую координату под базу 0 */
 asm    movdi,left;
 asm    dec di;
 /* вычислить смещение окна на странице */
 asm    shl di,1;
 asm    adddl,ax;

 /* прибавить смещение текущей страницы */
 asm    adddl,es:[44eh];
 /* временно сохранить смещение в AX */
 asm    movax,di;

 asm    moves,cx; /* адрес видеосегмента в ES */

 /* загрузить far-указатель на строку-источник */
 asm    lds si,sourcetext;
```

```
/* вычислить высоту окна в BX */
asm     movbx,bottom;
asm     sub bx,top;
asm     inc bx;
/* вычислить ширину окна в DX */
asm     movdx,right;
asm     sub dx,left;
asm     inc dx;
/* очистить direction-флаг для движения в
   сторону увеличения адреса */
asm     cld;
asm     jmp short skip_addition;

mem2video_row:  /* цикл пересылки строк окна */
/* вычислить смещение следующей строки окна и
   передать в DI */
asm     addax,160;
asm     movdi,ax;
skip_addition:
asm     movcx,dx;   /* ширина окна в CX */
/* пересылать символы с атрибутами на экран */
asm     rep movsw;
/* уменьшить счетчик строк окна на 1 */
asm     dec bx;
/* если счетчик не равен 0, то возврат в цикл */
asm     jne mem2video_row;

asm     pop ds; /* восстановить значение DS */
} /* restore_text() */

/********** Вставить символ в строку **********/
/* Функция вставляет переданный символ в
   указанную строку, сдвигая остальные символы
   строки вправо. Параметры:
   sourcestring - указатель на ту позицию в строке,
         в которую должен быть вставлен символ;
   stringlength - длина строки, считая от
           указанной позиции;
   character - код вставляемого символа.
*/
void insert_char(char far *sourcestring,
        int stringlength, int character)
{
asm     push    ds;    /* сохранить значение DS */

/* загрузить far-указатель на строку */
asm     lds si,sourcestring;
/* вычислить указатель на конец строки (0) */
asm     addsi,word ptr stringlength;
asm     push    ds;    /* скопировать DS в ES */
asm     pop es;
/* вычислить новое смещение для конца строки
   в DI */
asm     movdi,si;
asm     inc di;
/* вычислить количество сдвигаемых символов */
asm     movcx,stringlength;
asm     inc cx;
/* установить direction-флаг для движения в
   сторону уменьшения адреса */
asm     std;
asm     rep movsb;  /* перемещать символы */
/* загрузить в AL код вставляемого символа */
asm     moval,byte ptr character;
asm     stosb; /* вставить символ в строку */

asm     pop ds; /* восстановить значение DS */
} /* insert_char() */

/********** Удалить символ из строки **********/
/* Функция удаляет символ из строки, сдвигая
   оставшиеся символы строки влево. Параметры:
   sourcestring - указатель на тот символ строки,
          который необходимо удалить;
   stringlength - длина строки, считая от
          указанного символа.
*/
void delete_char(char far *sourcestring,
            int stringlength)
{
asm     push    ds;    /* сохранить значение DS */

/* загрузить far-указатель на строку */
asm     lds si,sourcestring;
asm     push    ds;    /* скопировать DS в ES */
asm     pop es;

asm     movdi,si; /* скопировать смещение в DI */
/* смещение следующего символа в SI */
asm     inc si;
/* счетчик сдвигаемых символов */
asm     movcx,stringlength;
/* очистить direction-флаг для движения в
   сторону увеличения адреса */
asm     cld;
asm     rep movsb;  /* перемещать символы */

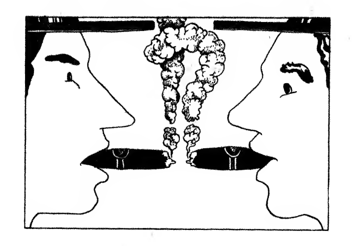
asm     pop ds; /* восстановить значение DS */
} /* delete_char() */

/************ Скопировать строку **************/
/* Функция копирует строку-источник в строку-
   приемник, возвращая при этом число
   скопированных символов (длину строки).
   Параметры:
   deststring - указатель на строку-приемник;
   sourcestring - указатель на строку-источник.
*/
int string_copy(char far *deststring,
        char far *sourcestring)
{
asm     push    ds;    /* сохранить значение DS */
/* загрузить far-указатель на строку-источник */
asm     lds si,sourcestring;
/* загрузить far-указатель на строку-приемник */
asm     les di,deststring;
asm     xor dx,dx;  /* счетчик в DX равен 0 */
/* очистить direction-флаг для движения в
   сторону увеличения адреса */
asm     cld;
nextchar:       /* цикл пересылки символов */
asm     lodsb;     /* загрузить символ в AL */
/* записать символ в строку-приемник */
asm     stosb;
/* увеличить счетчик символов на 1 */
asm     inc dx;
/* если не конец строки (0), то возврат в цикл */
asm     and al,al;
asm     jne nextchar;

/* уменьшить счетчик символов на 1, т.к. был
   сосчитан конец строки, и передать в AX */
asm     dec dx;
asm     movax,dx;
asm     pop ds; /* восстановить значение DS */
} /* string_copy() */

/* Конец файла INLUTIL.C */
```

*А. Синев*



*Этот выпуск "Между прочим..." посвящен в основном тем случаям, когда возникает необходимость дисассемблирования и анализа кода (попробуйте найти нужный участок кода в паре мегабайт дисассемблера!), и нужно уметь решить задачу исправлением пары байт. Чаще всего это требует довольно обширных знаний, включающих номера и параметры функций DOS и BIOS, структуру многочисленных управляющих блоков и т.д. и т.п. Тем, кого больше интересуют аппаратные средства, и адресованы заметки по поводу повышения производительности компьютеров и существующих разновидностей модемов.*

# МЕЖДУ ПРОЧИМ...

### Несколько тонкостей DOS

Как известно, при обращении к функции 013h прерывания 02Fh (DOS Multiplex Spooler Interrupt) в регистры ES:BX возвращается адрес, на который указывало прерывание 013h до загрузки DOS. При более тщательном анализе выяснилось, что, во-первых, указатель на "старое" тринадцатое прерывание возвращается также и в регистры DS:DX, а во-вторых, эта функция не запрашивает информацию о прерывании 013h, а заменяет старый обработчик тринадцатого прерывания на новый, адрес которого следует загрузить в эти пары регистров.

Уточним, что здесь речь идет именно об обработчике тринадцатого прерывания, к которому будет обращаться DOS при его выполнении, а не о текущем указателе на тринадцатое прерывание в таблице векторов.

Несколько слов еще об одном, весьма интересном адресе в ROM BIOS, который справедлив для огромного числа машин (если не для всех, хоть как-то совместимых с IBM PC). Было проверено несколько машин класса AT, XT, "Мазовия", "Искра" — для всех результат положительный. Итак: адрес прерывания 040h (аналог прерывания 013h, функции те же, но обслуживает только накопители на гибких дисках) равен F000:EC59.

### Строчная русская "Р" в Norton Commander

В приведенной ниже таблице указано, что нужно изменить в программах, входящих в комплект Norton Commander, чтобы они начали понимать прописную русскую букву "р".

| Версия | Программа | Смещение | Было | Записать |
|---|---|---|---|---|
| 2.0 | NC.EXE | $9CBD_{16}$ | $E0_{16}$ | $00_{16}$ |
| 3.0 | NCMAIN.EXE | $1E02_{16}$ | $E0_{16}$ | $00_{16}$ |
| 3.0 | WPVIEW.EXE | $36BE_{16}$ | $E0_{16}$ | $00_{16}$ |
| 3.0 | PARAVIEW.EXE | $8BE6_{16}$ | $E0_{16}$ | $00_{16}$ |
| 3.0 | DBVIEW.EXE | $2852_{16}$ | $E0_{16}$ | $00_{16}$ |

### Проблемы совместного использования Norton Disk Doctor и Disk Manager

При использовании NDD из-за конфликтов с Disk Manager обычно возникают два типа ошибок: Boot Record Program is Invalid и Invalid Disk Table in the Boot Record.

Первую ошибку исправить достаточно просто: используя Norton Utility необходимо переписать оригинальный загрузочный сектор исправляемого диска в файл, скопировать на исправляемый диск загрузочный сектор из загружаемого раздела винчестера, а затем восстановить таблицу параметров из сохраненного файла (таблица параметров диска находится в байтах загрузочного сектора, начиная с 0Bh и до байта 01Dh включительно).

Метод исправления второй ошибки пока неизвестен, но возникает она лишь в разделах с номером, превышающим 3. Это связано с тем, что при анализе раздела NDD использует таблицу разделов винчестера, однако при этом он не понимает нестандартных таблиц Disk Manager (для разделов 5-16). Поэтому номера последнего цилиндра, дорожки, головки у NDD неожиданно оказываются равными нулю, вместе с размером кластера в байтах (хотя размер кластера в секторах верный). Попытка отремонтировать такой диск с помощью NDD приведет к тому, что он будет испорчен почти безвозвратно. Пока в таком случае можно рекомендовать использовать программу CHKDSK, которая не анализирует таблицу разделов.

Знание структуры загрузочного сектора может сослужить хорошую службу и пользователям программы Advanced Disk Manager. У них появляется возможность, изменяя размер кластера (и, соответственно, размеры FAT) в загрузочном секторе, оптимизировать использование дискового пространства — чем меньше будет кластер, тем меньше места теряется из-за дискретности выделения дискового пространства.

## Режим Turbo в пакете LapLink

В меню выбора режимов пакета межмашинной связи LapLink есть не очень понятная опция Turbo Copy Mode. Что же она ускоряет и как работает?

При использовании этого режима скорость передачи данных действительно увеличивается, и весьма значительно — на 50-70%. При передаче информации через последовательный порт используется протокол, в котором последовательность передаваемых символов содержит не только саму передаваемую информацию, но и служебные биты, определяющие начало и конец отдельных слов и позволяющие проверить достоверность полученного байта (для этого служит бит проверки четности). Поэтому при передаче информации со скоростью 9600 бод реальная скорость передачи данных несколько меньше (на 20-40% в зависимости от конкретного протокола).

Кроме того, чтобы облегчить последовательному каналу различение отдельных слов, между ними делается довольно большая пауза. Дело в том, что при попытке передать слово до готовности принимающего порта к его приему, это слово будет утрачено. Большинство протоколов отслеживает такую ситуацию, выдавая сообщение о NAK (negative acknowledge — отсутствии подтверждения приема). Однако платы последовательного интерфейса, как правило, работают достаточно стабильно, что позволяет свести эту паузу почти к нулю без значительного увеличения количества сбойных байтов.

Именно это делается в режиме Turbo Copy. Кроме того, есть еще один фактор, использование которого может помочь повысить скорость передачи данных. Дело в том, что через последовательный порт данные передаются блоками фиксированной длины. Если какой-либо блок принят с ошибками, то он передается вновь и вновь. В режиме Turbo длина блока увеличивается с тем, чтобы передавать относительно меньший объем служебной информации.

Появление на экране сообщений об отсутствии подтверждения приема при работе в таком режиме говорит о том, что соединяемые платы работают с заметно разными (для компьютера) скоростями, что вызывает передачу слова до готовности принимающего компьютера к приему. Такое нередко случается при организации связи между компьютерами классов XT и AT/386. Данная проблема решается просто — достаточно выключить Turbo Copy Mode.

То же самое можно сделать с помощью опции Turbo Copy в режиме LINK пакета Norton Commander.

## Опять о быстродействии компьютеров

Некоторые пользователи считают, что замена микропроцессора с тактовой частотой 10 МГц на процессор с частотой 16 МГц означает немедленное увеличение производительности компьютера на 60%. Однако это далеко не так.

Разумеется, для того, чтобы ускорить работу машины, обязательно нужно увеличить частоту тактового генератора, так как именно он задает длительность машинного такта, который и определяет скорость работы системы. Придется повысить быстродействие и оперативной памяти, заменив микросхемы на более быстрые. Но именно здесь все и начнется... Может оказаться недостаточным быстродействие тех узлов, о существовании которых пользователь даже не подозревал (например, контроллер шины или схемы управления микросхемами динамического ОЗУ). Поэтому, если вы заинтересованы в более высокой производительности, то лучше заменить материнскую плату или даже сменить компьютер.

## Модемы: встроенный и внешний?

Чем отличается встроенный модем от внешнего? Какой из них выбрать?

Основное различие — в конструкции: встроенный модем представляет собой плату, монтируемую в разъем расширения компьютера, а внешний оформляется в виде отдельного устройства в небольшом корпусе вне компьютера. Кроме того, встроенный модем питается от источника питания компьютера, а внешний — от собственного блока питания.

На передней панели внешнего модема расположен ряд светодиодов, индицирующих режимы его

работы и состояние телефонной линии. У встроенного модема их нет, так как он скрыт внутри компьютера.

Более существенное отличие заключается в том, что внешний модем требует для своего подключения наличия свободного последовательного порта, что не всегда возможно. Встроенный модем, напротив, содержит все схемы последовательного интерфейса, поэтому может быть установлен вместо любого последовательного порта, за исключением тех, которые уже есть на вашем компьютере.

Принципы кодирования и декодирования информации одинаковы для модемов обоих типов. Для программ связи эти модемы также идентичны. Таким образом, принципиальной разницы между ними нет, поэтому используйте тот тип модема, который кажется вам более удобным.

*И.Вязаничев, О.Липкина, И.Свиридов*

По материалам журнала PC/Computing и бюллетеня "Софтпанорама".



# CASE - современная технология проектирования программного обеспечения

При создании больших программных комплексов, содержащих десятки тысяч кодов, почти невозможно избежать ошибок. Большая часть из них (60 — 70%) появляется на этапах анализа требований и создания структурной модели проекта, остальные — на этапе кодирования. Причем, на этапах отладки и тестирования программного обеспечения (ПО) обнаруживается около 75% ошибок кодирования и лишь 30% ошибок, внесенных на двух первых этапах создания ПО.

Для повышения надежности ПО и уменьшения числа ошибок было разработано множество методологий проектирования различного рода ПО — например, структурное программирование, ER-программы, диаграмма Бахмана. Все они используют диаграммную технику для графического представления предметной области и взаимосвязей объектов, потоков данных и т.д. Практика показывает, что если коллектив разработчиков придерживается какой-либо методологии и поддерживает графическое представление проекта, то существует большая вероятность разработки продукта высокого качества. Однако поддержка графического представления проекта в процессе проектирования вручную требует невероятных усилий, так как при внесении изменений в проект приходится целиком перечерчивать диаграммы и изменять описания, которые могут занимать сотни страниц. Массовое появление ПК и рабочих станций с развитой графикой дало возможность автоматизировать процессы отрисовки и документирования программ и значительно облегчить процесс их модификации. Эта возможность была реализована в виде большого числа коммерческих CASE-систем, появившихся на рынке в последнее время. Некоторые из них даже обеспечивают автоматическую генерацию программ по схеме предметной области для определенных (преимущественно АСУ) классов систем.

CASE (Computer Aided Software Engineering) технология представляет собой осуществление вполне естественной идеи автоматизации проектирования разработки ПО по примеру САПР в области машиностроения. Выигрыш от применения CASE-технологии особенно заметен при создании АСУ, СОД и ПО САПР. Средства CASE позволяют значительно уменьшить или вообще исключить многие проблемы разработки ПО и обойти узкие места в создании проекта. Отличие систем CASE от других инструментальных средств (электронные таблицы, СУБД, генератор приложений и т.п.) заключается в том, что эти системы на основе описаний требований к проектируемой системе создают интегрированную информационно-логическую модель (ИЛМ) системы. Основным приемом в CASE-технологии является разделение создания структурной схемы прикладной программы и программного кода. Это позволяет проектировщику ПО сосредоточиться в большей степени на разработке архитектуры системы, чем на создании кода.

Появление CASE-индустрии и постоянно повышающийся интерес к ней вполне закономерны, так как все фирмы-разработчики коммерческого ПО (ОС, СУБД, компиляторов и т.п.) давно в той или иной мере ис-

пользуют элементы CASE-технологии. Ряд из них можно классифицировать как CASE-средства нижнего уровня, например, языково-чувствительные редакторы, системы тестирования и управления исходными кодами и т.д.

Для того, чтобы более строго определить понятие CASE-технологии, рассмотрим процесс разработки и сопровождения коммерческого программного обеспечения.

Цикл разработки ПО можно разбить на следующие этапы:
— анализ требований к проекту;
— создание структурной модели проекта;
— создание выполняемого кода;
— тестирование программы;
— реализация и поддержка работающей программы.

На этапе анализа требований выделяются и исследуются требования заказчика к создаваемому ПО, которые затем преобразуются в спецификации требований. Этот документ формирует требования обычно в специфическом формате, который разбивает каждое требование на простые детали. Существуют стандартные промышленные форматы, например, DOD-STD-2167 или IEEE, применяемые при создании спецификаций требований. Кроме пронумерованного списка особенностей и возможностей создаваемого ПО, этот документ содержит описание требований к интерфейсу пользователя, перечень специальных средств технического и программного обеспечения (ОС, системы БД и т.д.) и критерии оценки эффективности.

На этапе создания структурной модели проекта спецификации требований преобразуются в архитектурную схему проекта, детализирующую предполагаемое исполнение ПО. В основе такого преобразования лежит принцип организации ПО в виде совокупности независимых модулей, каждый из которых может быть сконструирован, построен и отлажен отдельно. При этом каждый модуль представляется другому "черным ящиком" с точно определенными входами и выходами. На этапе построения структурной модели проектная задача разбивается на отдельные модули, организуется иерархия модулей, определяются пути обмена данными между модулями, структуры этих данных, главные подпрограммы в каждом модуле (их заголовки, входы и выходы, возвратные коды и коды ошибок), форматы внешних файлов данных и пути доступа к ним, конструируются ключевые алгоритмы. Выходом на данном этапе является графическая схема создаваемого ПО.

Исходя из целей облегчения всех этапов этого цикла, можно дать следующие общие характеристики средствам CASE:

— устранение сложности путем разбивки требований и архитектурной схемы проекта на простые, управляемые компоненты;

— удешевление процесса разработки ПО по сравнению с традиционными методами;

— обеспечение контроля за правильностью взаимодействия программных модулей и полнотой определения структур данных;

— графическое ориентирование: средства CASE имеют возможность представлять спецификации и конструкторскую документацию визуально;

— синхронизация изменений на всех этапах проекта.

Существует свыше 35 производителей CASE-средств. Из них около 15 предлагают средства, основанные на методе структурного анализа (Yourdon/De Macro). Большинство CASE-методологий основано на одних и тех же идеях обработки диаграмм потоков данных. Все они используют графическое представление, во всех, в том или ином виде, организована иерархия. Большинство из них изображает связи и структуры при помощи прямоугольных рамок или "пузырьков", соединенных линиями, и многие имеют средства для выборочного показа деталей структуры. Действительно ли много различий между существующими методологиями?

Например, диаграммы потоков данных, диаграммы Warnier-Orr, Scandura FLOWforms и многие структурные карты выполняются с использованием "Outline Processor metaphor", позволяющего расширять и сужать объект за счет показа дополнительных деталей или их удаления с экрана. Диаграммы W-O и диаграммы Jackson Structured Programming изображают иерархическую структуру программы. По большому счету, единственная разница между ними в том, что диаграммы W-O используют строчные метафоры (скобки), а Jackson — диаграммы-метафоры-рамки.

Несколько производителей CASE-средств, такие как Nastec, Visual Software, Future Tech, усиливают свои продукты, обеспечивая возможность настройки и объединения с новыми, определяемыми пользователем, методологиями. Эти средства обеспечивают возможность создания новых экранных объектов и определения правил связи между ними. Рассмотрим некоторые конкретные CASE-системы.

## Пакет DesignAid
## Фирма Nastec

Это инструментальное средство с высокой функциональной мощностью и достаточно простым интерфейсом. Подготовка к работе с пакетом состоит из следующих этапов: создание проектной директории; построение словаря проекта (с помощью отдельной утилиты); запуск системы. Войдя в систему, пользователь может вести проектирование одним из множества способов: можно начать работу с определения структур, либо с создания диаграмм и связи их различных уровней и т.д. Возможностей очень много.

Центральная идея DesignAid — работа с вложенными файлами. Файлы, которые логически связаны между собой, могут быть открыты один из другого (вложены) и отображены на экране. Например, пользователь может иметь диаграмму, в которой изображены несколько процессов; каждый из них, в свою очередь, может быть связан с другими диаграммами или файлами. Возможно, что один файл будет простым процес-

сом или текстовым описанием процесса, другой — подробной диаграммой потоков данных с показом "родительских связей".

Существует реальная необходимость в использовании вложенных файлов. Они позволяют переходить с одного уровня диаграммы на другой, просто пересекая границы файлов. Убрав курсор из текста файла, можно снова вернуться на прежний уровень. С помощью такого механизма DesignAid обеспечивает средство, позволяющее легко, быстро, логично перемещаться с уровня на уровень в рамках проекта. DesignAid также позволяет вставлять вложенные файлы непосредственно в текущий файл или диаграмму. Для этого достаточно написать имя файла и заключить его в скобки. Слово в скобках DesignAid воспринимает как ссылку на файл. Благодаря этому можно очень просто читать вложенные файлы. Например, для того, чтобы прочитать текстовый файл, связанный с отдельным объектом диаграммы, необходимо нажать всего две клавиши.

Одно из преимуществ взятых в скобки файлов заключается в том, что с их помощью можно создавать суммарные файлы (или меню-файлы), т.е. списки, содержащие взятые в скобки имена файлов проекта, сопровождаемые комментариями о том, что эти файлы делают. Такой меню-файл может быть легко открыт и просмотрен, необходимо только выбирать из него пункты и их просматривать. Меню-файлы можно также использовать в качестве легкого средства группировки файлов. К такому меню-файлу можно применять общие для группы файлов операции.

Цена пакета: 6900 долл.

## Пакет Exelerator Фирма Index Technology

Успех данного пакета обусловлен в первую очередь тем, что он был одним из первых пакетов CASE для IBM PC. Конечно, существует и множество других причин, по которым его использует большинство организаций, работающих с CASE.

Exelerator — хорошо сбалансированный пакет, обеспечивающий высококачественные инструментальные средства для поддержки фаз анализа и проектирования цикла жизни системы (SDLC). Из достоинств пакета можно выделить хорошие средства построения диаграмм, описывающих фазы анализа и проектирования; мощные средства проверки правил структурной методологии; хорошие средства создания прототипов отчетов и экранов; прекрасные графические возможности и возможности генерации документации. Более того, благодаря успеху Exelerator на рынке многие разработчики CASE предусмотрели в своих пакетах средства для осуществления интерфейса (загрузки/выгрузки) с пакетом Exelerator.

При входе в Exelerator пользователь должен указать свой идентификатор и пароль, а затем выбрать необходимый проект из ранее определенного им пользовательского списка. Доступ к отдельным режимам Exelerator (графические работы, проектирование экранов и отчетов, анализ, изменение параметров, установка Exelerator, работа со словарем XDL, создание документации) осуществляется из главного меню системы. Пакет работает с оборудованием, клавиатурой, мышью и дисплеем достаточно эффективно и согласованно. Несмотря на то, что Exelerator предоставляет пользователю большой набор функций, управление пакетом через систему меню и экранов единого формата несложно.

Цена пакета: 8400 долл.

## Пакеты IEW/Analisis Workstation, IEW/Design Workstation Фирма Knowledge Ware

Большим успехом на рынке CASE пользуется система Information Engineering Workbench (IEW) фирмы Knowledge Ware. Она представляет собой группу пакетов, предназначенных для рабочих станций на базе IBM PC. Пакеты выполняют работу по планированию, анализу, проектированию и разработке программного обеспечения. Фирма Knowledge Ware также продает генератор кода GAMMA для большой ЭВМ. В его состав входят центральный репозитарий, который заполняется информацией из пакетов инструментальных средств, работающих на персональном компьютере. В совокупности пакеты системы IEW поддерживают все этапы жизненного цикла прикладной системы за исключением этапа эксплуатации.

Пакеты IEW/Analisis Workstation и IEW/Design Workstation являются частями системы IEW версии 5.0 и стоят каждый по 8625 долл. Все пакеты IEW используют интерфейс GEM, что позволяет сбалансированно применять экран, клавиатуру и мышь. В начале работы с пакетами IEW пользователь должен указать свой идентификатор и пароль. После их проверки на экране появляется рабочая область, а в верхней части экрана — меню в стиле Microsoft Windows или Macintosh. Использование при работе с IEW меню и пиктограмм значительно облегчает работу.

Одним из основных преимуществ IEW являются его средства анализа и проектирования, предоставляющие хорошие возможности для построения диаграмм, необходимых для моделирования как данных, так и процессов проектирования базы данных и процедур. Кроме того, IEW позволяет одновременно просматривать и модифицировать несколько диаграмм или объектов. Для каждой диаграммы можно создать свое окно под управлением GEM-интерфейса. Использование множества окон особенно полезно при создании или изменении объекта в репозитарии системы — Encyclopedia. Обычно при этом необходимо использовать информацию о других объектах, которые уже внесены в репозитарий.

IEW удобно использовать для логического моделирования данных и физического проектирования баз данных. IEW/Design Workstation работает с диаграммами структур данных и диаграммами баз данных, основан-

ными на реляционной и иерархической моделях, а также на модели плоских файлов. На основе диаграмм для иерархической модели могут быть сгенерированы описания на языке DLI для СУБД IMS. На основе диаграмм для реляционной модели могут быть сгенерированы описания на языке SQL для СУБД DB2. Поскольку пакеты IEW/Design и IEW/Analisis являются частями единой системы IEW, они хорошо увязаны по данным с другими пакетами системы, осуществляющими планирование и разработку (программирование) прикладной системы.

Цена пакета IEW/Analisis Workstation: 8625 долл.

Цена пакета IEW/Design Workstation: 8625 долл.

### Пакет POSE (Picture Oriented Software Engineerign) Фирма Computer Systems Advisors

Этот пакет, представляющий собой средство графического программирования, появился на рынке год назад. Он был создан в Сингапуре и привезен в 1988 г. в США. Пакет стоит недорого при цене каждого из его модулей немногим более 500 долл. По выполняемым функциям все подсистемы POSE можно разбить на три группы: средства моделирования данных и проектирования базы данных; средства моделирования и проектирования процессов; средства построения матричных диаграмм и прототипов. Организация может приобрести отдельные модули пакета за 595 долл. или группы из четырех модулей, объединенные в "инструментальное средство", за 1195 долл.

В состав инструментального средства для моделирования данных Data Midel Toolkit входят: средство построения диаграмм моделей данных, нормализатор моделей данных, средство построения диаграмм логической структуры БД и "помощник" для работы с БД. В состав инструментального средства для моделирования процессов Prosess Model Toolkit входят средства построения: диаграмм потоков данных, диаграмм декомпозиции, структурных карт и карт действий. Инструментальное средство для построения прототипов экранов и отчетов и средство для построения матричных диаграмм продаются отдельно по 595 долл. каждое.

Интерфейс пользователя с пакетом POSE осуществляется через главное меню, учитывающее конфигурацию POSE для каждого конкретного пользователя. После выбора необходимой функции из меню происходит загрузка программы для выполнения этой функции, и в верхней части экрана появляется соответствующее меню. Все модули POSE используют однотипные меню и единую терминологию. Следует заметить, что средства моделирования, проектирования данных и процессов, создания прототипов, проектирования и генерации БД имеют достаточно высокое качество исполнения и высокую функциональную мощность.

При создании в POSE новых объектов и сущностей пользователю предоставляются специальные экраны для их описания. Например, для процессов, обозначенных в диаграмме потоков данных, можно указать идентификаторы, имена, входы и выходы, логические операторы. Логические операторы образуют что-то вроде миниспецификаций. Каждый модуль POSE может формировать отчеты о созданных с его помощью объектах, а каждое средство построения диаграмм имеет свой генератор отчетов, позволяющий создавать стандартные отчеты и отчеты по индивидуальным требованиям пользователей. Все модули могут выполнять функции поиска объектов/сущностей, печати диаграмм и сохранения/восстановления.

### Заключение

Каковы же дальнейшие перспективы развития CASE-средств? В идеале эти средства должны были бы по требованиям пользователя создавать свободное от ошибок прикладное ПО. На самом деле это невозможно, поскольку всегда требуется проведение тестирования и экспертизы ПО для выявления технических промахов и ошибок. Поэтому прогресс в области CASE-технологий ожидается прежде всего на пути расширения круга средств, позволяющих генерировать исполняемые коды автоматически. Некоторые средства уже на существующем уровне развития CASE-технологий способны по спецификациям требований генерировать исполняемые коды, а в течение следующих пяти лет процент автоматически сгенерированных прикладных программ достигнет, по некоторым оценкам, уровня 40-75%, в зависимости от области приложения.

Основные направления совершенствования CASE-средств, по-видимому, следующие:

— улучшение технологий миниспецификаций;

— автоматическая генерация кодов по миниспецификациям;

— возможность повторного использования существующих программных модулей;

— интеграция с другими пакетами ПО;

— возможность сопровождения выполняемых модулей.

*О.Посудина, М.Ривкин, А.Смородинский*

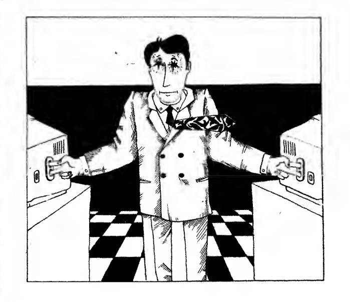
**По материалам:**

A.Fisher "CASE. Using Software Development Tools", 1989.

M.Chen "CASE present status and future direction", DATA BASE, vol.20, №1 1989.

E.Yourdon "DEC CASE. American programmer", 1990.

R.Davis "What the Real World Is Saying About CASE", Chiff Information Officer Journal, vol. 2, 1990.



# СЕТЬ простыми средствами

Эта история началась в тот момент, когда один из студентов пытался установить плату контроллера накопителя на магнитной ленте во включенный компьютер, что привело, как и следовало ожидать, к короткому замыканию. Результатом этого незапланированного "эксперимента" стали вышедшие из строя платы контроллера на МЛ и контроллера винчестера. К счастью, ничего больше сгореть не успело, так как защита блока питания вовремя отключила напряжение. Таким образом, из персонального компьютера "MAZOVIA — CM1914" типа IBM PC XT с винчестером емкостью 20 Мбайт и двумя гибкими дисками по 360 Кбайт получилась персоналка с двумя гибкими дисками, что даже хуже, чем ЕС-1840, у которой, как известно, два дисковода по 720 Кбайт.

Те, кто работал на машинах без винчестера, понимают, насколько сложно реализовать весь процесс написания программ на такой машине. Тем более, что основной системой программирования в нашей организации является Clipper, работа с которой в таких условиях превращается в настоящую пытку.

Наличие же рядом с рабочим местом еще одной персоналки типа IBM PC AT еще больше "действовало на нервы" тому, кто работал на "MAZOVIA".

В создавшейся ситуации возникло естественное желание попытаться каким-либо образом связать между собой два компьютера с тем, чтобы использовать ресурсы и, прежде всего, винчестер AT совместно. Соединение двух компьютеров по COM-портам не составило большого труда. Для этого мы взяли три обычных телефонных провода длиной метров по пять (растояние между машинами), припаяли к ним разъемы и соединили два порта по приведенной ниже схеме (рис.1).

Из схемы видно, что второй контакт на одном компьютере соединялся с третьим контактом на другом компьтере и наоборот. Контакты "земля" соединялись между собой. Остальные контакты разъемов оставались свободными.

Подбор программного обеспечения занял достаточно много времени, так как оно должно было позволить обращаться к винчестеру AT-машины (сервер) из любой программы, запускаемой на XT (сателит), обеспечивая при этом обслуживание запросов на AT-машине в фоновом режиме с тем, чтобы на ней мог работать второй программист.

Первым делом испытанию подверглись коммуникационные возможности Norton Commander. Однако, несмотря на высокую скорость обмена файлами между компьютерами (115200 бод), необходимые требования не выполнялись. Связь существовала только тогда, когда на обеих машинах был загружен NC (обратиться с сателита на сервер можно только из NC). При этом сервер полностью подчинен сателиту — выполнять какие-либо действия на сервере не представлялось возможным.

Следующим шагом в наших поисках были программы типа D-link, Lap-Link, Lap-Link Plus и другие. Хотя эти программы и позволяли обращаться с сателита на сервер из любой программы, при этом на сервере не обеспечивался фоновый режим работы и он был занят исключительно обслуживанием другой машины. Попытка исправить положение с помощью пакета DESQView успехом не увенчалась из-за недостатка оперативной памяти на AT (Ext = 384 Кбайта).

Некоторые из испытанных пакетов были очень требовательны к другим программам и драйверам, загруженным в память компьютера — например, отказывались работать с Advance Disk Manager, воспринимая только раздел С: и игнорируя все остальные.

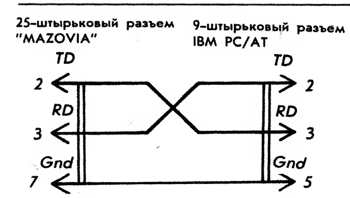



Рис.1. Кабель для соединения компьютеров

Наш окончательный выбор пал на пакет LanLink, который в достаточной мере удовлетворял нашим требованиям.

LanLink позволил соединить компьютеры в локальную сеть типа "звезда" через последовательные порты. При этом в каждой паре соединенных машин одна выступала в качестве сервера, а вторая — в качестве сателита.

В первоначальном варианте нами использовалась сеть из двух компьютеров. В настоящее время мы соединили между собой четыре машины, три из которых AT, а одна — "MAZOVIA".

Для обеспечения связи требуется по одному COM-порту для каждой связи. Следовательно, если вы хотите соединить два компьютера между собой так, чтобы каждый из них мог использовать ресурсы другого, необходимо иметь по два COM-порта на каждом из них.

В нашем случае на двух машинах имелось по одному порту. AT выполняла функциии сервера для XT. При этом на сателит (XT) загружались четыре драйвера (по одному на каждый раздел винчестера), объемом по 900 Байт каждый, и программа обслуживания сети — LanSat объемом 24 Кбайт. Общий объем программного обеспечения на сервере (AT) составил 64 Кбайта (24 Кбайта для запуска сервера и 40 Кбайт для обслуживания рабочей станции сети).

Скорость передачи данных по кабелю составляет 115200 бод (согласно документации), но такие операции, как копирование, осуществляются медленнее, чем при передаче с помощью опции LINK пакета Norton Commander. На XT затраты времени при работе с дисками сервера в несколько раз меньше, чем при работе с гибкими дисками, но значительно больше, чем при работе с винчестером.

Программное обеспечение достаточно надежно и практически не вступает в конфликт с другими драйверами и программами, в частности, с утилитой 800 и системой ADM.

Нам удалось найти всего несколько программ, которые не смогли работать при загруженной сети. При этом программное обеспечение сервера оказалось намного более чувствительным к такого рода программам, чем программное обеспечение сателита. Например, популярная программа Side Kick "вешает" сервер после своего запуска. Программа фонового форматирования дискет ConFormat также не может работать — она "вешает" сервер после завершения процесса форматирования. Препятствуют работе AT-машине в режиме файл-сервера и различные игрушки.

На сателите некоторые программы не воспринимают path при задании файлов (например PkZip) и работают только в текущем каталоге.

При работе на сервере работа периферийной машины практически не ощущается. Время компиляции программ Clipper-ом почти не изменяется. Увеличение компиляции программ Clipper-а на периферийной машине незначительно по сравнению с компиляцией на локальном винчестере.

Но эксплуатация сети из двух машин выявила и недостатки данного варианта программного обеспечения. В частности, при работе на сервере с гибкими дисками, особенно в режиме копирования или, что еще хуже, форматирования, существенно снижается время реакции сервера на запрос сателлита. И если процесс, связанный с использованием гибких дисков, достаточно длительный, то работать на сателите невозможно. Кроме того, вследствие недостатков программного обеспечения в этих случаях на экране сателита иногда появляется сообщение "Server communication error. Retry or Abort?". После ответа "R" процесс работы сети продолжается нормально.

"Зависание" сервера в процессе передачи данных приводнт в большинстве случаев к невозможности выполнения текущей программы на периферийной машнне, и управление передается операционной системе. Но, если сервер "завис" не в процессе передачи данных, то после перезагрузки сервера программа на сателите может продолжить свою работу. "Зависание" периферийной машины на состоянии сервера обычно не отражается.

Сеть может быть легко снята с помощью популярной программы Release, а затем снова загружена, когда она потребуется периферийной машине.

В настоящее время к уже подключенным компьютерам добавились еще две AT с двумя COM-портами (в отличие от предыдущих, у которых было по одному COM-порту). Их соединение в сеть осуществлялось по схеме, приведенной на рис.2.

Как видно из приведенной схемы, XT имеет в своем распоряжении винчестер первого сервера и через него (так как он является сателитом для другого сервера) еще и винчестер второго сервера. Таким образом, все машины имеют доступ к винчестеру главного сервера, что дает возможность держать на нем основное программное обеспечение, освободив диски других машин для хранения данных и часто используемого программного обеспечения.

Эксплуатация приведенной схемы показала, что время доступа к винчестеру второго сервера через первый сервер значительно (в несколько раз) больше времени доступа к винчестеру первого сервера. Заметно увеличилось количество сообщений "Server communi-

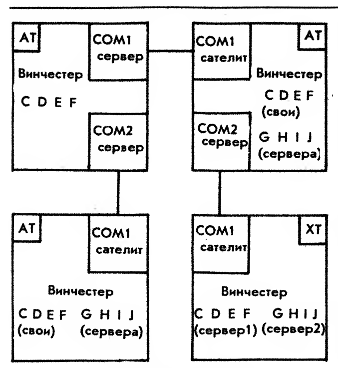



Рис.2. Структурная схема сети.

cation error", особенно на AT-сателитах, что объясняется недостатками программного обеспечения LanLink. Очевидно, разработчики сети предполагали, что в качестве рабочих станций будут использоваться в основном XT-машины. Работа сети в такой конфигурации начинает ощущаться и на главном сервере как некоторое увеличение времени реакции на нажатие клавиш.

Основываясь на практическом опыте, мы выработали некоторые рекомендации по структуре программного обеспечения в сети и принципам работы. Во-первых, программное обеспечение, которое используется лишь периодически, держать на всех машинах нецелесообразно. Его лучше держать на сервере, загружая по сети при необходимости. Это освобождает винчестер для других данных. Во-вторых, желательно снизить до минимума интенсивность обмена по сети с тем, чтобы иметь возможность отключить сеть в любой момент (если, например, не хватает памяти или необходимо выполнить программу, которая конфликтует с сетью). В-третьих, на файл-сервере желательно не работать с гибкими дисками, так как при этом сильно снижается скорость обмена по сети. Это имеет особое значение для промежуточного сервера.

Небольшое время эксплуатации не позволило нам выявить все достоинства и недостатки сети LanLink. Мы не успели опробовать работу ее для сетевых пакетов, для разрабатываемых автоматизированных систем управления производством, для разделения других ресурсов (принтеры), но мы надеемся, что эта небольшая статья поможет вам решить некоторые проблемы.

*В.Камышин, О.Романов*

Материал получен при содействии редакции бюллетеня "Софтпанорама"

*Обучающий курс журнала LAN Magazine представляет собой серию статей по вопросам локальных сетей для начинающих пользователей. В этом курсе в простой и доступной форме излагаются основные концепции, лежащие в основе организации локальных сетей. Каждый месяц в сборнике КомпьютерПресс будет печататься очередной выпуск серии, посвященный какому-либо вопросу, связанному с организацией локальных сетей. Вырезайте и сохраняйте выпуски серии и вы сможете получить в конце курса брошюру, которая будет представлять собой введение в локальные сети.*
*В этом выпуске мы продолжим обсуждение вопросов, связанных с сетевыми возможностями операционной системы OS/2.*

# Локальные сети от А до Я:

## курс обучения

### *ЧАСТЬ 19. ВЗАИМОДЕЙСТВИЕ МЕЖДУ ПРОЦЕССАМИ В OS/2*

В предыдущем выпуске обучающего курса был начат разговор о сетевых возможностях операционной системы OS/2 — в частности, о многозадачном режиме работы OS/2 и его использовании в локальных сетях для организации многопользовательской среды. Мы отметили, что многозадачность значительно расширяет сетевые возможности OS/2 по сравнению с такой операционной системой, как DOS. Однако OS/2 обладает еще одной особенностью, существенной для работы в локальных сетях, а именно, — организацией взаимодействия между процессами.

**Коротко о взаимодействии между процессами**

Существует множество форм взаимодействия между процессами. Пользователи Microsoft Windows и Apple Macintosh наверняка знакомы с одним из наиболее простых типов взаимодействия, с помощью которого можно копировать информацию из одной области экрана в другую. Так, если в одном из окон Microsoft Windows или Apple Macintosh работает электронная таблица, требуемые данные можно "вырезать" и перенести в другое окно, в которое загружен какой-либо текстовый редактор. Операционная система OS/2 также поддерживает данный тип обмена информации между задачами, однако ясно, что в среде локальной сети необходима поддержка и более сложных процессов взаимодействия.

В OS/2 имеется целый ряд так называемых интерфейсов прикладных программ (API), которые позволяют программистам создавать программы, взаимодействующие между собой при параллельном выполнении на компьютере в среде OS/2. Более того, с помощью интерфейсов прикладных программ можно разделить одну задачу на несколько частей, которые называются процессами, и обеспечить взаимодействие между ними. Преимущество такого деления в том, что оно обеспечивает возможность параллельного прогона нескольких блоков одной и той же программы, что может существенно ускорить ее выполнение.

Наиболее существенным достижением является поддержка интерфейсами прикладных программ OS/2 совместного использования результатов одного процесса другими, что создает основу для вы-

полнения на одном компьютере программ типа "клиент-сервер". Для наглядности представим, что на компьютере выполняется сортировка базы данных, в которой хранится информация о сотрудниках учреждения (например, их имена и фамилии, адреса, номера телефонов и размер зарплаты). Назовем программу сортировки сервером базы данных. Затем на том же компьютере запустим еще две программы: генератор отчетов и программу вывода на экран телефонной книги с адресами и номерами телефонов всех сотрудников. Благодаря возможностям интерфейсов прикладных программ OS/2 и генератор отчетов, и программа вывода телефонной книги на экран могут одновременно получить необходимую информацию от сервера базы данных. Такой подход позволяет существенно упростить процесс разработки прикладных программ типа генераторов отчетов, телефонных или адресных книг и т.п., поскольку отпадает необходимость в написании больших и сложных программ обработки данных.

Однако это еще не все! Программа вывода на экран телефонной книги может быть запущена на одном компьютере, генератор отчетов — на другом, а сервер базы данных — на третьем, и все три программы будут работать так, как если бы это было на одной машине. Именно такая организация вычислительного процесса является характерной особенностью специализированных сетевых приложений.

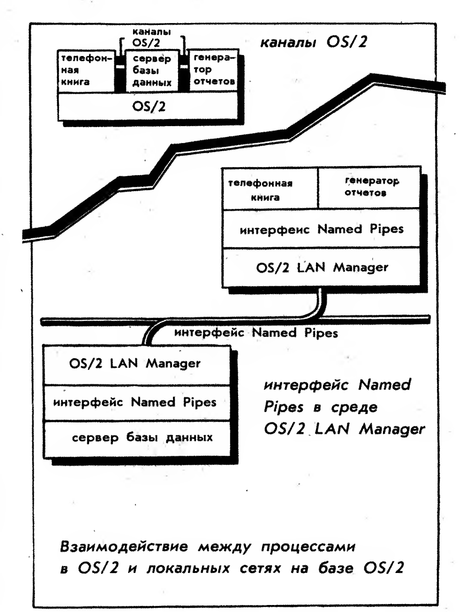
## Взаимодействие между процессами в OS/2

Рассмотрим, каким же образом осуществляется взаимодействие между процессами в OS/2. Для этих целей предусмотрено пять механизмов: общая память, флажки, семафоры, очереди и конвейеры.

Общая память (shared memory). Представляет собой однн из наиболее естественных способов взаимодействия между процессами через общую область памяти. В этом случае на системе OS/2 приходится разрешать конфликты при одновременном обращении нескольких процессов к общей памяти. Общая память является основной формой взаимодействия между процессами. Все остальные формы строятся на ее базе.

Флажки (flags). Флажки или сигналы (signals) действуют подобно прерываниям. При выставлении соответствующего флажка одним процессом, его выполнение прерывается и управление передается другому процессу, "отвечающему" за обработку данной ситуации. Флажки чаще всего используются как средства реагирования на экстренные ситуации, возникающие в системе.

Семафоры (semaphores). Они подобны флажкам, но в отличие от последних предназначены не для обработки определенных ситуаций, а для синхронизации обмена данными — например, могут быть использованы для координирова-




Взаимодействие между процессами в OS/2 и локальных сетях на базе OS/2

ния работы процессов с общей памятью. Существуют системные и локальные семафоры. Системные чаще всего выполняют синхронизацию задач в различных процессах, а локальные — в рамках одного процесса.

Очереди (queues). Используются для синхронизации и обмена данными между процессами. Один процесс может занести данные в очередь, а другой (владелец очереди) выбрать их из очереди. Однако принцип "первым пришел — первым обслужен" в такой очереди не действует, поскольку процесс-владелец может выбирать данные в любом порядке.

Каналы (pipes). Подобны очередям, за исключением того, что они работают по принципу "первым пришел — первым обслужен" (FIFO). Один процесс может открыть канал и передать по нему данные другому (принимающему) процессу.

Очереди и каналы являются наиболее важными механизмами поддержки взаимодействий между процессами в OS/2, особенно в среде локальных сетей. Одно из важнейших преимуществ очередей и каналов OS/2 — это простота их использования. Программист "открывает" очередь или канал, а затем считывает и записывает в них данные точно так же, как он делает это при работе с файлами.

Как уже было отмечено в предыдущем выпуске обучающего курса, OS/2 не является многопользовательской операционной системой. Многозадачность и взаимодействие между процессами OS/2 реализуются только для программ, выполняемых на одном персональном компьютере. Однако при использовании соответствующей сетевой операционной системы эти средства OS/2 могут быть распределены на рабочие станции локальной сети для поддержки многопользовательских приложений.

## Взаимодействие между процессами в локальных сетях

Можно смело сказать, что ключевыми элементами при использовании OS/2 в локальных сетях являются интерфейсы прикладных программ, обеспечивающие взаимодействие между процессами. Из множества таких интерфейсов следует отметить три наиболее важных для локальных сетей: NetBIOS, APPC и Named Pipes.

Интерфейс NetBIOS представляет собой базовый протокол для программ IBM PC LAN Program фирмы IBM и MS-Net фирмы Microsoft. Многие программы, обеспечивающие работу локальных сетей, особенно средства электронной почты и шлюзы для связи с универсальными ЭВМ, были написаны с использованием этого интерфейса прикладных программ.

Интерфейс APPC был разработан фирмой IBM и первоначально предназначался для обеспечения связи между универсальными и миди-ЭВМ. Сейчас IBM использует этот интерфейс при реализации взаимодействия между персональными компьютерами, работающими в локальной сети. К сожалению, прикладных программ для этого интерфейса было написано немного.

Named Pipes является самым новым из рассматриваемых интерфейсов. Он был разработан фирмой Microsoft в качестве базы для сетевой операционной системы MS-OS/2 LAN Manager и расширения каналов OS/2.

Рассмотренные интерфейсы различаются, главным образом, размерами занимаемой памяти и простотой использования. Так, APPC более "интеллектуален", чем Named Pipes или NetBIOS, однако занимает больше памяти. При разработке задач взаимодействия между процессами в Named Pipes требуется написание значительно меньшего количества кодовых строк, чем в APPC или NetBIOS. Это происходит потому, что Named Pipes уже содержит большую часть логики, необходимой для поддержки взаимодействия между двумя процессами. Мощные логические средства в Named Pipes значительно упрощают разработку распределенных прикладных программ типа "клиент-сервер", по сравнению с аналогичными программами на базе интерфейсов APPC и NetBIOS. Более того, такая "встроенность" большей части сетевой логики позволяет интерфейсу Named Pipes повысить производительность всей сети, поскольку в процессе выполнения сетевых приложений количество сообщений между процессами значительно уменьшится, а это, в свою очередь, означает уменьшение сетевого трафика.

Можно предположить, что большая часть новых приложений для локальных сетей на базе OS/2 будет написана на базе интерфейса Named Pipes. В настоящее время основные фирмы-поставщики сетевых операционных систем или уже поддерживают Named Pipes, или объявили об обеспечении поддержки этого интерфейса в ближайшем будущем.

Подводя итоги описания сетевых возможностей операционной системы OS/2, можно сказать, что многозадачный режим работы и поддержка взаимодействия между процессами делают OS/2 незаменимым инструментарием при разработке многопользовательских распределениых сетевых приложений.

*В.Демидов*
*В.Миропольский*

**По материалам:**
"LAN tutorial series", LAN Magazine, October 1989.

# IBM PC
# для пользователя

## ЧАСТЬ 1
## НАЧАЛЬНЫЕ СВЕДЕНИЯ

### Глава 1. Устройства, входящие в IBM PC

Персональный компьютер IBM PC включает следующие устройства (рис. 1.1):

— процессор, выполняющий управление компьютером, вычисления и т.д.;

— клавиатура, позволяющая вводить символы в компьютер;

— монитор (или дисплей) для изображения текстовой и графической информации;

— накопители (или дисководы) для гибких магнитных дисков, используемые для чтения и записи на магнитные диски (дискеты);

— накопитель на жестком магнитном диске, предназначенный для чтения и записи на съемный жесткий магнитный диск (винчестер).

Кроме того, к компьютеру могут подключаться:

— принтер — для вывода на печать текстовой и графической информации;

— мышь — устройство, облегчающее ввод информации в компьютер;

— джойстик — манипулятор для игр;

— а также другие устройства.

Более подробно об устройствах Вы узнаете, когда прочтете эту главу до конца. Однако сначала мы расскажем, как надо включать и выключать компьютер.

#### Включение и выключение компьютера

**Проверка величины напряжения сети.** Перед первым включением компьютера следует проверить, соответствует ли напряжение в сети тому, на которое рассчитан компьютер (многие компьютеры могут работать при нескольких значениях входного напряжения, например, при 220 и 110 вольтах). При необходимости надо установить переключатель напряжения на компьютере в правильное положение.

**Стабилизация напряжения.** Во многих населенных пунктах СССР напряжение в сети может сильно колебаться. Для компьютера такие изменения напряжения являются нежелательными (особенно вредны резкие понижения напряжения), поэтому лучше подключать компьютеры через стабилизаторы. Лучше всего использовать специальные стабилизаторы для компьютеров, которые не только обеспечивают строго постоянное напряжение питания, но и дают возможность работы компьютеров при полном отключении электропитания в интервале от 15 мин. до нескольких часов. За это время можно, во всяком случае, полностью завершить ведущиеся на компьютере работы, чтобы при его выключении не произошло потери информации. Бытовые стабилизаторы таким свойством не обладают, но их применение может быть весьма полезно.

**Включение компьютера.** Для включения компьютера необходимо:

— включить стабилизатор напряжения, если компьютер подключен через стабилизатор напряжения;

— включить принтер(если он нужен);

— включить компьютер (переключателем на корпусе компьютера);

— включить монитор компьютера.

После этого на экране компьютера появятся сообщения о ходе работы программ проверки и начальной загрузки компьютера. Когда начальная загрузка операционной системы будет закончена, появится приглашение операционной системы, например

C:\> или 20:59 C;\WORK>

(вид приглашения может меняться пользователем с помощью команды DOS prompt). Появление приглашения означает, что операционная система готова к приему команд.

**Выключение компьютера.** Для выключения компьютера надо:

— закончить работающие программы;

— ввести команду PARK (и нажать клавишу [Enter]) для установки головок чтения-записи на жестком диске в положение, при котором можно безопасно выключать электропитание;

---

*Главы из второго, переработанного и дополненного издания книги В.Э. Фигурнова "IBM PC для пользователя", М., совместное издание "Финансы и статистика" и агентства "КомпьютерПресс", 1991.*

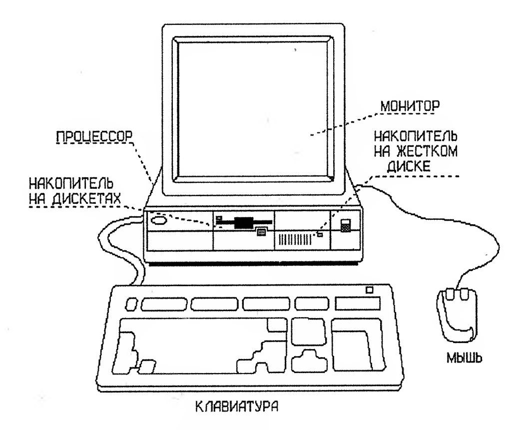



**Рис. 1.1. Общий вид персонального компьютера IBM PC**

— выключить принтер (если он включен);

— выключить монитор компьютера;

— выключить компьютер (переключателем на корпусе компьютера);

— выключить стабилизатор, если компьютер подключен через стабилизатор напряжения.

Ниже рассказывается о назначении, функциях и возможностях различных устройств, входящих в компьютер IBM PC или подключаемых к нему.

## Процессор

Процессор персонального компьютера IBM PC содержит:

*основной микропроцессор,* управляющий работой компьютера и выполняющий все вычисления;

*оперативную память,* в которой располагаются программы, выполняемые компьютером, и используемые программами данные. Емкость оперативной памяти, как правило, равна 640 Кбайтам (байт — единица информации, эквивалентная восьми двоичным цифрам или одному символу из 256 возможных);

*электронные схемы (контроллеры),* управляющие работой различных устройств, входящих в компьютер (монитора, накопителей на магнитных дисках и т.д.);

*порты ввода-вывода,* через которые процессор обменивается данными с внешними устройствами. Имеются специализированные порты, через которые происходит обмен данными с внутренними устройствами компьютера, и порты общего назначения, к которым могут подсоединяться различные дополнительные внешние устройства (принтер, мышь и т.д.). Порты общего назначения бывают двух видов: параллельные (обозначаемые LPT1—LPT4) и асинхронные последовательные (обозначаемые COM1—COM3). Параллельные порты выполняют ввод и вывод с большей скоростью, чем последовательные, но требуют и большего числа проводов для обмена данными.

Основной микропроцессор определяет быстродействие компьютера. Исходный вариант компьютера IBM PC и модель IBM PC XT используют микропроцессор Intel-8088. Сейчас распространен вариант этого микропроцессора с увеличенной производительностью (тактовой частотой). Для обозначения того, что модель компьютера имеет повышенную производительность (тактовую частоту), к названию модели компьютера иногда прибавляют "Turbo", например Turbo-XT.

Модель Turbo-XT имеет приблизительно в 1,7 раза большую производительность по сравнению с моделью IBM PC XT.

Модель IBM PC AT использует более мощный микропроцессор Intel-80286 и ее производительность приблизительно в 5-6 раз больше, чем у IBM PC XT. Имеются модели IBM PC AT с повышенной производительностью, скорость их работы в 1,5-2 раза больше, чем у IBM PC AT. Микропроцессор Intel-80286 имеет несколько больше возможностей по сравнению с Intel-8088, но подавляющее большинство производителей программного обеспечения не используют дополнительных возможностей Intel-80286, чтобы их программы могли работать и на IBM PC XT (исключений немного, например, Windows-286 фирмы Microsoft).

Модели серии PS/2, как правило, используют мощный микропроцессор Intel-80386 (впрочем, имеются модели этой серии, работающие с Intel-80286 и даже с Intel-8088). Производительность моделей компьютеров с Intel-80386 приблизительно в 3-4 раза больше, чем у IBM PC AT. Однако это увеличение производительности существенно в основном для решения задач, требующих больших вычислений или обработки данных большого объема информации, так как для основного круга применений персональных компьютеров модели IBM PC AT (и Turbo-AT) обеспечивают достаточно высокую производительность.

Микропроцессор Intel-80386 имеет значительно больше возможностей по сравнению с Intel-8088. Это 32-разрядный микропроцессор, т.е. он может обрабатывать 4-байтовые целые числа и адреса. Поэтому многие производители программного обеспечения разрабатывают версии своих программ для Intel-80386 или программы специально для Intel-80386.

Фирмой Intel разработан также микропроцессор Intel-80386-SX, который хотя и немного дороже Intel-80286, но и обладает теми же возможностями, что Intel-80386, только с более низким быстродействием. Поэтому сейчас многие производители компьютеров предпочитают использовать в своих машинах не Intel-80286, а Intel-80386-SX, что позволяет увеличить быстродействие и дает возможность пользователям работать с программным обеспечением, рассчитанным специально на Intel80386.

Микропроцессор Intel-80386 (вариант Inboard-386) может быть вставлен в компьютеры IBM PC XT и IBM PC AT вместо микропроцессоров Intel-8088 и Intel-80286. Это позволяет при умеренных затратах значительно увеличить скорость работы компьютера.

Микропроцессор Intel-80486 ничем не отличается от Intel-80386, но его производительность в 3-4 раза выше.

**Математические сопроцессоры.** Микропроцессоры Intel-8088, Intel-80286 и Intel-80386 не содержат специальных команд для работы с числами с плавающей точкой. При необходимости проведения расчетов с числами с плавающей точкой каждая операция над этими числами моделируется с помощью нескольких десятков операций микропроцессоров Intel-8088/80286/80386. Это сильно снижает эффективность применения компьютера для научных вычислений, при использовании машинной графики и для других применений с интенсивным использованием чисел с плавающей точкой. Поэтому в этих случаях следует использовать компьютеры IBM PC с установленным математическим сопроцессором Intel-8087, Intel-80287 или Intel-80387. Этот сопроцессор может быть установлен на любую модель компьютера IBM PC, что увеличивает скорость выполнения операций с плавающей точкой в 5-15 раз. Микропроцессор Intel-80486 сам поддерживает операции с плавающей точкой, поэтому при его использовании математический сопроцессор не требуется.

**Основной микропроцессор и диапазон применимости компьютера.** Быстродействие основного микропроцессора и его скорость обмена данными с другими устройствами определяют диапазон применения компьютера. В настоящее время компьютеры типа IBM PC XT на основе микропроцессоров Intel-8088 или Intel-8086 выходят из употребления в деловых применениях. Эти компьютеры уже рассматриваются в качестве бытовых, поскольку их быстродействие во многих часто встречающихся приложениях является недостаточным. Наибольшее распространение имеют компьютеры типа IBM PC AT с тактовой частотой от 10 до 16 МГерц, они обеспечивают достаточное быстродействие в большинстве приложений. Когда требуется обработка больших объемов информации или проведение больших расчетов, например, при обработке видеосигналов или для управления локальной сетью персональных компьютеров, используются компьютеры на основе Intel-80386. Если и этот микропроцессор не обеспечивает требуемого быстродействия, можно использовать Intel-80486.

## Оперативная память

Оперативная память компьютера IBM PC с процессором Intel-8088 или Intel-8086 (например, IBM PC XT) может иметь размер более 1 Мбайта, поскольку эти микропроцессоры могут обращаться не более чем к 1 Мбайту памяти. Эта память состоит из двух частей. Первые 640 Кбайт памяти могут использоваться прикладными программами и операционной системой. Остальная память используется для служебных целей:

— для хранения части операционной системы DOS, которая обеспечивает тестирование компьютера, начальную загрузку операционной системы, а также выполнение основных низкоуровневых услуг ввода-вывода;

— для передачи изображения на экран;

— для хранения различных расширений операционной системы, которые поставляются вместе с дополнительными устройствами компьютера.

Как правило, когда говорят об объеме оперативной памяти компьютера, то имеют в виду именно первую ее часть, которая может использоваться прикладными программами и операционной системой. Мы также будем в дальнейшем поступать таким образом.

Первые модели IBM PC, появившиеся в начале 80-х годов, часто имели небольшой размер оперативной па-

мяти — 256 или 384 Кбайта. В настоящее время стоимость оперативной памяти стала гораздо ниже, и поэтому почти все персональные компьютеры имеют размер оперативной памяти 640 Кбайт, самое меньшее — 512 Кбайт.

Микропроцессоры Intel-80286 и Intel-80386 уже могут обращаться с оперативной памятью большего размера (первый — с 16 Мбайтами, а второй — с 4 Гбайтами). Однако режим, в котором они могут это делать (так называемый "защищенный" или "protected" режим), не совместим с программами, работающими под управлением DOS. Поэтому использование оперативной памяти свыше 640 Кбайт в рамках операционной системы DOS не может осуществляться непосредственно. Для доступа к добавочной оперативной памяти разработаны специальные программы ("драйверы"), которые можно вызывать из прикладных программ. Эти драйверы получают запрос от прикладной программы (например, на пересылку блока данных из добавочной памяти в обычную или наоборот), переходят в "защищенный" режим работы микропроцессора, выполняют запрос и переключаются обратно в обычный режим работы микропроцессора.

### Накопители на гибких магнитных дисках (дискетах)

Гибкие диски (дискеты) позволяют переносить документы и программы с одного компьютера на другой, хранить информацию, не используемую постоянно на компьютере, делать архивные копии информации, содержащейся на жестком диске. Чаще всего на компьютере имеются два дисковода для дискет. Работа на компьютере с одним дисководом для дискет имеет некоторые особенности.

**Дискеты размером 5,25 дюйма.** Наиболее распространены дискеты размером 5,25 дюйма (133 мм, рис 1.2). В настоящее время чаще всего используются дискеты емкостью 360 Кбайт (обозначаемые Double Side/Double Density, DS/DD) и 1,2 Мбайта (Double Side/High Detsity, DS/HD). Иногда встречаются дискеты прежних лет выпуска, имеющие меньшую емкость либо рассчитанные для использования на дисководах с одной головкой (односторонние дискеты).

Для записи и чтения дискет емкостью 1,2 Мбайта предназначены специальные накопители, которые устанавливаются на компьютерах моделей IBM PC AT и PS/2. Эти накопители могут также читать дискеты емкостью 360 Кбайт, но информация, записанная ими на такие дискеты, плохо считывается на дисководах для дискет емкостью 360 Кбайт.

Дисководы для дискет емкостью 1,2 Мбайта снаружи никак не отличаются от дисководов для дискет емкостью 360 Кбайт. Однако используемая в них техника записи на дискеты различна: в дисководах емкостью 1,2 Мбайта используются головки чтения-записи, обеспечивающие более узкую дорожку для записи информации. Дискеты емкостью 1,2 Мбайта имеют специальное магнитное покрытие, которое позволяет записывать на них эту узкую дорожку информации. Это магнитное покрвтие труднее намагнитить и размагнитить, чем обычное, и поэтому такие дискеты не могут использоваться в дисководах емкостью 360 Кбайт. Как правило, на дискетах емкостью 360 Кбайт вокруг внутреннего отверстия имеется темное кольцо, а дискеты емкостью 1,2 Мбайта — нет. Кроме того, дискеты емкостью 1,2 Мбайта имеют более темное магнитное покрытие. Это позволяет в сомнительных случаях различать дискеты разной емкости.

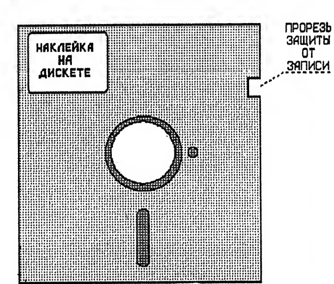



Рис. 1.2. Дискета размером 5,25 дюйма (133 мм)

Дискеты размером 5,25 дюйма требуют бережного обращения: их надо хранить в бумажном конверте, аккуратно вставлять в дисковод. Нельзя также гнуть и трогать руками открытые участки магнитного покрытия. Следует не допускать попадания пыли на дискету.

**Дискеты размером 3,5 дюйма.** В компьютерах серии PS/2 и в портативных компьютерах часто используются накопители на дискетах размером 3,5 дюйма (89 мм) и емкостью 0,7 и 1,4 Мбайта (рис. 1.3). Эти дискеты заключеиы в жесткий пластмассовый конверт, что значительно повышает их надежность и долговечность.

**Защита дискет от записи.** На дискетах размером 5,25 дюйма имеется порезь для защиты от записи (см.рис. 1.2.). Если эту прорезь заклеить, то на дискету нельзя будет произвести запись (разумеется, при условии, что дисковод исправен).

На дискетах размером 3,5 дюйма вместо прорези защиты от записи имеется специальный переключатель, разрешающий или запрещающий запись на дискету.

**Инициализация (форматирование) дискет.** Перед первым использованием дискеты необходимо специальным образом инициализировать. Это делается с помощью программы DOS Format.

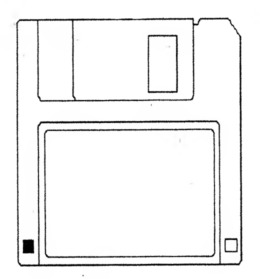


**Рис. 1.3. Дискета размером 3,5 дюйма (89 мм)**

## Накопители на жестком диске

Накопители на жестком диске (винчестеры) предназначены для постоянного хранения информации, используемой при работе с компьютером: программ операционной системы, часто используемых пакетов программ, редакторов документов, трансляторов с языков программирования и т.д. Наличие жесткого диска значительно повышает удобство работы с компьютером.

**Емкость диска.** Для пользователя накопители на жестком диске отличаются друг от друга прежде всего своей емкостью, т.е. тем, сколько информации помещается на диске. На моделях IBM PC XT жесткий диск чаще всего имеет емкость 20 Мбайт, а на IBM PC AT — 40 Мбайт. Бывают жесткие диски большей емкости — 80, 160, 300 Мбайт.

**Скорость работы диска.** Вторая существенная для пользователя характеристика диска — время доступа к информации. Для областей применения, требующих интенсивного обмена с дисками (например информационных систем), этот показатель является исключительно важным. Разумеется, обеспечение высокой скорости доступа требует значительных усилий при разработке дисков, поэтому "быстрые" диски стоят значительно выше обычных.

З а м е ч а н и е. В ранних версиях операционной системы MS DOS нельзя было непосредственно работать с дисками емкостью более 32 Мбайт. Это затрудняло использование дисков большой емкости, их приходилось логически "разбивать" на несколько частей (логических дисков) и работать с каждой из них как с отдельным диском, что не всегда приемлемо. Теперь эта проблема устранена. В версиях операционной системы MS DOS 4.00 и последующих, а также в операционной системе DR DOS можно использовать логические диски любого размера.

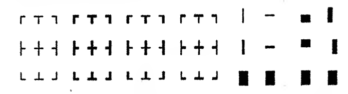
## Мониторы

Монитор (дисплей) компьютера IBM PC (см. рис. 1.1) предназначен для вывода на экран текстовой и графической информации. Мониторы бывают цветными и монохромными. Они могут работать в одном из двух режимов: текстовом или графическом.

**Текстовый режим.** В текстовом режиме экран монитора условно разбивается на отдельные участки — знакоместа, чаще всего — на 25 строк по 80 символов (знакомест). В каждое знакоместо может быть выведен один из 256 заранее заданных символов. В число этих символов входят большие и малые латинские буквы, цифры, символы

~!@#$%^&*()_+|`-=\
{}[]:;"'«,».?/

а также псевдографические символы, используемые для вывода на экран таблиц и диаграмм, построения рамок вокруг участков экрана и т.д.



В число символов, изображаемых на экране в текстовом режиме, могут входить символы кириллицы (буквы русского алфавита).

На цветных мониторах каждому знакоместу может соответствовать свой цвет символа и свой цвет фона, что позволяет выводить красивые цветные надписи на экран. На монохромных мониторах для выделения отдельных частей текста и участков экрана используются повышенная яркость символов, подчеркивание и инверсное изображение (темные символы на светлом фоне).

**Графический режим.** Графический режим монитора предназначен для вывода на экран графиков, рисунков и т.д. Разумеется, в этом режиме можно также выводить и текстовую информацию в виде различных надписей, причем эти надписи могут иметь произвольный шрифт, размер букв и т.д.

В графическом режиме экран монитора состоит из точек, каждая из которых может быть темной или светлой на монохромных мониторах или одного из нескольких цветов — на цветном. Количество точек по горизонтали и вертикали называется разрешающей способностью монитора в данном режиме. Например, выражение "разрешающая способность 640x200" означает, что монитор в данном режиме выводит 640 точек по горизонтали и 200 точек по вертикали. Следует заметить, что разрешающая способность не зависит от размера экрана монитора, подобно тому, как большой и маленький телевизоры имеют на экране 625 строк развертки изображения.

**Часто используемые мониторы.** Наиболее широкое распространение в компьютере IBM PC получили

мониторы типов MDA, CGA, Hercules, EGA и VGA. Их характеристики приведены в табл. 1.1.

**Таблица 1.1**

| Монитор | Цвет/ моно | Текстовый режнм | Графическнй режим |
|---|---|---|---|
| MDA | Монохромный | 80x25, 2 цвета | 640x200, 2 цвета |
| CGA | Цветной | 80x25, 16 цветов 320x200, 4 цвета | 640x200, 2 цвета |
| Hercules | Монохромный | 80x25, 2 цвета | 720x348, 2 цвета |
| EGA | Цветной | 80x25, 16 цветов 80x43, 16 цветов | 640x350, 16 цветов |
| VGA | Цветной | 80x25, 16 цветов 80x50, 16 цветов | 640x480, 16 цветов |

В настоящее время мониторы MDA и CGA используются уже относительно редко, так как они не обладают надлежащей разрешающей способностью, что приводит к быстрому утомлению глаз. Кроме того, они не имеют возможности программиой загрузки шрифтов символов, поэтому для изображения букв кириллицы в текстовом режиме приходится заменять электронные схемы, хранящие шрифты символов (знакогенераторы). Иногда, впрочем, можно ие заменять знакогенератор, а записать в него с помощью специальных приборов нужные шрифты символов.

Мониторы EGA и VGA в последнее время получили очеиь широкое распространение, оии фактически стали стандартом для тех применений, в которых требуются графические средства приемлемого качества. Монитор VGA имеет несколько большую стоимость, но он обеспечивает и более высокую разрешающую способность. Кроме того, в мониторе VGA расстояиия между соседними точками изображения по вертикали и по горизонтали совпадают, что значительно упрощает графические программы.

В некоторых применениях, иапример, в полиграфии или обработке изображеиий, требуются специальиые моииторы, дающие более высокое качество изображения, чем мониторы EGA и VGA. Например, в издательской системе полезно иметь монитор с экраном, на котором изображается вся печатаемая страница в масштабе 1:1. Такие специальные моииторы существуют, но они достаточио дороги и требуют специальиого программного обеспечения. Наиболее часто используются мониторы с разрешающей способностью 800x600 и 1024x768 точек.

**Скорость работы.** Важной характеристикой монитора и его адаптера является также скорость работы. В текстовом режиме все мониторы работают достаточно быстро, ио при выводе графических изображений с высокой разрешающей способиостью скорость работы может быть весьма существенна. В приложеииях с интенсивным использованием графики (обработке изображений, анимации, конструировании и т.д.) может оказаться необходимым использование "быстрого" монитора и адаптера. Разумеется, они стоят значительно дороже обычных.

## Клавиатура

Клавиатура IBM PC предназиачеиа для ввода в компьютер информации от пользователя. На рис. 1.4 показана модель клавиатуры для IBM PC AT, для других моделей компьютера расположение и число клавиш на клавиатуре может несколько отличаться, ио изиачение одииаковых клавиш на разных клавиатурах, разумеется, совпадает. На компьютерах типа IBM PC XT, иапример, функциоиальных клавиш ие 12, а 10 ([F1]—[F10]), и расположены они не в верхнем ряду клавиатуры, а слева. На персональных компьютерах советского производства надписи на клавишах могут быть сделаны по-русски. Кроме того, у персональиых компьютеров советского производства на клавиатуре имеются клавиши "РУС" и "ЛАТ" для переключения с русского шрифта на латинский и обратно.

Расположеиие латинских букв иа клавиатуре IBM PC, как правило, такое же, как на английской пишущей машинке, а букв кириллицы — как на русской пишущей машинке.

В дальнейшем изложеиии мы будем для обозначения клавиш употреблять надпись иа клавише стандартной клавиатуры, заключенную в квадратные скобки. Например, [Shift] обозначает клавишу с надписью "Shift".

**Ввод прописных и строчных букв.** Для ввода прописиых букв и других символов, располагающихся на верхнем регистре клавиатуры, имеется клавиша [Shift]. Например, чтобы ввести строчную букву "d", надо нажать клавишу, иа которой изображено"D", а чтобы ввести прописную букву "D", надо нажать клавишу [Shift] и, не отпуская ее, иажать иа клавишу "D".

Клавиша [Caps Lock] служит для фиксации режима прописных букв. Это удобно при вводе текста, состоящего из таких букв. Повторное иажатие клавиши [Caps Lock] отменяет режим прописных букв. В режиме [Caps Lock] нажатие клавиши [Shift] дает возможность ввода строчных букв. Иногда клавиша [Caps Lock] используется для других целей, иапример, для переключения иа русский алфавит.

**Переключение на другой алфавит.** Переключение клавиатуры с режима ввода латинских букв на режим ввода русских букв на персональиых компьютерах советского производства осуществляется с помощью клавиш "РУС" и "ЛАТ". На компьютерах зарубежного производства это переключение выполняется специальиой программой — драйвером клавиатуры. Эта программа, как правило, запускается в начале работы с компьютером и затем постоянно находится в оперативной памяти компьютера. Функции этой программы — воспринимать нажатия клавиш иа клавиатуре и передавать соответствующие символы операционной сис-

теме DOS. После нажатия определенной комбинации клавиш драйвер клавиатуры начинает передавать в компьютер символы другого алфавита. Одни драйверы используют для переключения на другой алфавит клавишу [Caps Lock], другие — одновременное нажатие обеих клавиш [Shift] или одновременное нажатие клавиш [Ctrl] и [Alt], бывают и другие способы переключения.

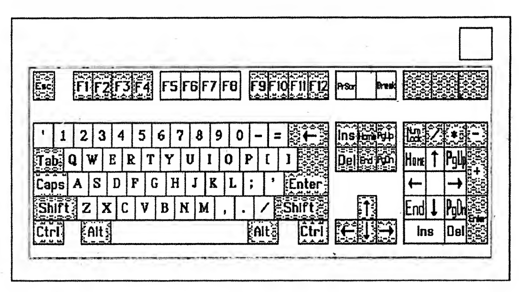



**Рис. 1.4. Клавиатура компьютера IBM PC (модель клавиатуры для IBM PC AT)**

**Специальные клавиши клавиатуры.** Кроме алфавитно-цифровых клавиш и клавиш со знаками пунктуации, на клавиатуре имеется большое число специальных клавиш.

Клавиша **[Enter]** (на некоторых клавиатурах — [Return] или [CR]) предназначена для окончания ввода строки. Например, при вводе команд DOS ввод каждой команды должен оканчиваться нажатием клавиши [Enter].

Клавиша **[Del]** (Delete — удаление) используется для удаления символа, находящегося под курсором.

Клавиша **[Ins]** (Insert — вставка) предназначена для переключения между двумя режимами ввода символов: ввода с раздвижкой символов (вставка) и ввода с замещением ранее набранных символов (замена).

Клавиша **[Backspace]** (стрелка влево над клавишей [Enter]) удаляет символ, находящийся слева от курсора.

Клавиши **[←]**, **[→]**, **[↑]**, **[↓]**, **[Home]**, **[End]**, **[PgUp]**, **[PgDn]** называют клавишами управления курсором. Как правило, нажатие на них приводит к перемещению курсора в соответствующем направлении или к "перелистыванию" изображаемого на экране текста.

Клавиша **[NumLock]** (блокировка цифр) включает и выключает режим, в котором при нажатии на клавиши [Home], [↑], [PgUp], [←], [→], [End], [↓], [PgDn], [Ins] и [Del], расположенные в правой части клавиатуры, в компьютер вводятся цифры 1—9, 0 и точка. Этот режим удобен для ввода чисел.

Клавиша **[Esc]** (Escape — убегать, спасаться), как правило, используется для отмены какого-либо действия, выхода из режима программы и т.д.

Функциональные клавиши **F1—F10** (на некоторых клавиатурах F1—F12) предназначены для различных специальных действий. Их действие определяется выполняемой программой.

Клавиши **[Ctrl]** и **[Alt]**. На клавиатуре имеются специальные клавиши [Ctrl] и [Alt]. Как и клавиша [Shift], они предназначены для изменения значений других клавиш. Клавиши [Ctrl] и [Alt] вводятся в комбинации с другими клавишами, и выполняющаяся программа может особым образом реагировать на такие комбинации клавиш.

Например, в описании программы может быть написано, что для выполнения определенного действия следует ввести (или нажать) [Alt—X]. Это означает, что пользователь должен нажать клавишу [Alt] и, не отпуская ее, нажать клавишу "X".

**Ввод в компьютер произвольных символов.** Клавиша [Alt] и клавиши [0]—[9], расположенные в правой части клавиатуры (т.е. клавиши перемещения курсора и клавиша [Ins]), дают возможность вводить в компьютер произвольные символы, в том числе и те, которых нет на клавиатуре. Для этого необходимо нажать клавишу [Alt], не отпуская ее, набрать десятичный код нужного символа на клавишах 0-9 в правой части клавиатуры, а затем отпустить клавишу [Alt].

**Особые комбинации клавиш.** Имеются комбинации клавиш, обрабатываемые специальным образом:

**[Ctrl—Break]** — завершение работы выполняемой программы или команды;

[Ctrl—Alt—Del] (одновременное нажатие клавиш [Ctrl], [Alt] и [Del] — перезагрузка DOS (см.ниже);

[Shift—PrtScr] — печать на принтере копии содержимого экрана;

[Ctrl—PrtScr] — включает и выключает режим копирования на принтер выводимой на экран информации;

[Ctrl—NumLock] — приостанавливает выполнение программ. Для продолжения их выполнения надо нажать любую клавишу. Для команд программ Dos аналогичную функцию выполняет [Ctrl—S];

[Ctrl—Alt—F1] — переключение на стандартную американскую клавиатуру (используется после команды KEYBxx);

[Ctrl—Alt—F2] — переключение на национальную клавиатуру (используется после команды KEYBxx).

При диалоге пользователя с DOS, например вводе команд DOS, могут употребляться следующие специальные комбинации клавиш.

[Ctrl—C] — прекращение работы любой команды или программы DOS.

[Ctrl—P] — включает и выключает режим копирования на принтер выводимой на экран информации.

[Ctrl—S] — приостанавливает выполнение программ.

[F6] — ввод символа конца файла (этот символ обозначается [Ctrl—Z] и имеет код 26).

[F7] — ввод символа с кодом 0 (зачем это нужно, я не знаю).

### Принтеры

Принтер (или печатающее устройство) предназначен для вывода информации на бумагу. Все принтеры могут выводить текстовую информацию, многие из них могут выводить также рисунки и графики, а некоторые принтеры могут выводить и цветные изображения.

Существует несколько тысяч моделей принтеров, которые могут использоваться IBM PC. Как правило, применяются принтеры следующих типов: матричные, струйные, литерные и лазерные.

**Матричные (или точечно-матричные) принтеры** — наиболее распространенный тип принтеров для IBM PC (рис. 1.5).

Принцип печати этих принтеров таков: печатающая головка принтера содержит вертикальный ряд тонких металлических стержней (иголок). Головка движется вдоль печатаемой строки, а стержни в нужный момент ударяют по бумаге через красящую ленту. Это и обеспечивает формирование на бумаге символов изображений.

В дешевых моделях принтеров используется печатающая головка с 9 стержнями. Качество печати у таких принтеров посредственное, но его можно несколько улучшить с помощью печати в несколько проходов (от двух до четырех). Более качественная и быстрая печать обеспечивается принтерами с 24 печатающими иголками (24-точечными принтерами). Бывают принтеры и с 48 головками, они обеспечивают еще более качественную печать. Скорость печати точечно-матричных принтеров — от 10 до 60 с на страницу.

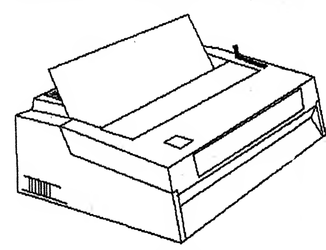


Рис. 1.5. Матричный принтер

**Струйные принтеры.** В этих принтерах изображение формируется микрокаплями специальных чернил, выдуваемых на бумагу с помощью сопел. Этот способ печати обеспечивает более высокое качество печати по сравнению с матричными принтерами, он очень удобен для цветной печати. Однако струйные принтеры дороже матричных и требуют более тщательного ухода и обслуживания. Скорость печати струйных принтеров приблизительно такая же, как у матричных, — от 10 до 60 с на страницу.

**Литерные принтеры** обеспечивают высокое качество печати, но набор печатаемых символов у них ограничен. У наиболее распространенных моделей таких принтеров количество символов в наборе недостаточно для печати текстов с русскими и английскими буквами. Кроме того, такие принтеры не могут выводить графическую информацию.

Скорость печати литерных принтеров невысока — от 15 мин до 20 с на страницу.

**Лазерные принтеры** обеспечивают в настоящее время наилучшее (близкое к типографскому) качество печати. В этих принтерах для печати используется принцип ксерографии: изображение переносится на бумагу со специального барабана, к которому электрически притягиваются частички краски. Отличие лазерного принтера от обычного ксерокопировального аппарата состоит в том, что печатающий барабан электризуется с помощью лазера по командам из компьютера.

Лазерные принтеры, хотя и достаточно дороги (приблизительно в 2—4 раза больше стоимости IBM PC XT), являются наиболее удобными устройствами для получения качественных печатных документов. Разрешающая способность лазерных принтеров, как правило, 300 точек на дюйм. Скорость печати лазерных принтеров — от 15 до 3 с на страницу при выводе текстов. Страницы с рисунками могут выводиться

значительно дольше, на вывод рисунков может потребоваться до десяти минут.

**Потребительские качества принтеров.** Перечислим основные качества принтеров, определяющие их сравнительные достоинства с точки зрения пользователя.

*Качество и скорость печати* — обеспечивает ли принтер необходимое качество печати, и если да, то с какой скоростью. Так, одни принтеры (например, лазерные) могут печатать качественные документы со скоростью 5 с на страницу, а другие (например, 9-точечные матричные принтеры) требуют около 5 мин для печати страницы текста с качеством, близким к качеству пишущей машинки.

*Наличие русских букв* — имеется ли у данного принтера возможность печати русских букв, и если да, то как она обеспечивается.

а) Шрифты русских букв могут иметься в самом принтере (в его постоянном запоминающем устройстве). В этом случае после включения принтер сразу готов к печати текстов с русскими буквами. Если коды русских букв в принтере такие же, как в компьютере, русские тексты могут печататься так же, как и английские, например, командами DOS print или copy. Если же коды русских букв в принтере не такие, как в компьютере, то для печати русских текстов необходимо использовать специальные программы. В этом случае либо для печати файла, содержащего текст с русскими буквами, необходимо запустить программу печати этого файла, либо в начале работы с компьютером запускается постоянно находящаяся в оперативной памяти (резидентная) программа, которая перекодирует все русские буквы, посылаемые на принтер, в соответствующие им коды принтера. Последний вариант, как правило, более удобен, хотя он уменьшает емкость оперативной памяти, доступной другим программам.

б) Шрифты русских букв могут загружаться с помощью программ. В этом случае перед печатью русских текстов необходимо запустить специальную программу для загрузки шрифтов русских букв. При выключении принтера (а при сбоях во время печати иногда приходится выключать и затем включать принтер) шрифты русских букв пропадают из его памяти, и необходимо снова запустить программу для загрузки шрифтов русских букв.

как и в случае (а), желательно, чтобы коды русских букв в принтере были такие же, как в компьютере, так как при этом для печати русских текстов не требуется использование специальных программ.

в) Русские буквы могут печататься только в графическом режиме, т.е. так же, как печатаются рисунки. Печать в графическом режиме дает возможность получения любых шрифтов, однако печать при этом, как правило, в несколько раз медленнее, чем в обычном (текстовом) режиме. Печать текстов может выполняться специальными программами или редакторами документов.

*Надежность* — какова надежность принтера при печати типичных документов и при работе с имеющейся у пользователя бумагой.

*Возможность автоматической подачи бумаги* при печати на отдельных листах бумаги. Если такой возможности принтер не предоставляет, то при печати на отдельных листах бумаги кто-то должен стоять у принтера и вставлять листы бумаги в принтер.

*Шрифты* — какие шрифты букв поддерживает принтер. Некоторые принтеры предоставляют большое количество (иногда несколько десятков) шрифтов, а некоторые — только один. Количество шрифтов, которые может поддерживать лазерный принтер, зависит от его типа и от объема имеющейся в нем памяти (см. ниже).

*Смена красящих элементов* — какова продолжительность работы принтера с данной красящей лентой (кассетой красящего порошка или чернил), обеспечивается ли их замена. Многие пользователи, не имея сменных кассет с красящей лентой, заменяют красящую ленту в кассете на ленту для пишущей машинки, предварительно сварив ее в кольцо с помощью паяльника, или же печатают без красящей ленты через копирку. Замена красящего порошка для лазерного принтера или специальных чернил для струйного принтера с помощью таких "домашних средств" невозможна, поэтому следует обеспечивать поставку этих красящих элементов.

*Совместимость с имеющимися программами.* Различные принтеры имеют различные команды для управления сменой шрифтов, движением бумаги, рисованием графических изображений и т.д. Как правило, прикладные программы обеспечивают работу только с наиболее распространенными типами принтеров. Поэтому желательным качеством принтера является его совместимость по управляющим командам с наиболее распространенными принтерами, например, с матричными принтерами Epson или IBM Graphics, с лазерными принтерами Hewlett-Packard или языком описания страниц PostScript.

**Особенности лазерных принтеров.** Среди лазерных принтеров имеются два основных типа: Совместимые с HP LaserJet фирмы Hewlett-Packard и "понимающие" язык PostScript, разработанный фирмой Adobe. Бывают и такие принтеры, которые не "понимают" ни языка LaserJet, ни языка PostScript, но тогда вместе с ними обычно поставляются программы, эмулирующие LaserJet или PostScript. Эмуляция, как правило, замедляет печать на принтере в несколько раз, особенно при выводе рисунков.

Принтеры, понимающие язык PostScript, обычно раза в полтора дороже, чем эквивалентные по производительности принтеры типа LaserJet. Однако они имеют и больше возможностей: они могут выводить буквы любых размеров (скажем, кегля 9,5, если это понадобится), инвертировать буквы и т.д. Поэтому при применении компьютеров в качестве настольных издательских систем более целесообразно использовать принтеры типа PostScript. В остальных случаях достаточно иметь принтер типа LaserJet.

Очень важной характеристикой лазерного принтера является объем имеющейся в нем оперативной памяти. Эта оперативная память используется для построения выводимых на принтер рисунков и для хранения загружаемых в принтер шрифтов символов. Принтеры с небольшим объемом оперативной памяти (от 0,5 до 1 Мбайта) не могут выводить большие рисунки (скажем, во всю страницу) и хранить достаточное количество шрифтов. Поэтому для серьезной издательской работы требуется больший объем оперативной памяти принтера.

### Другие устройства, подключаемые к компьютеру

Ниже кратко описываются другие устройства, которые могут подключаться к компьютеру IBM PC. Разумеется, это перечисление является далеко не полным, оно приведено для общего представления о возможностях использования персональных компьютеров.

**Мышь** — манипулятор для ввода информации в компьютер. Название "мышь" это устройство получило за свой внешний вид (см. рис. 1.1) и способ его использования. Мышь представляет собой небольшую коробочку (обычно серого цвета) с двумя или тремя клавишами, легко умещающуюся в ладони. Вместе с проводом для подключения к компьютеру это устройство действительно напоминает мышь с хвостом. Чтобы изменить положение курсора на экране монитора, пользователь перемещает мышь по столу или другой ровной поверхности. Когда необходимо выполнить то или иное действие, например, выполнить пункт меню, на который указывает курсор, пользователь нажимает ту или иную клавишу мыши. Некоторые прикладные программы рассчитаны на работу с мышью, но большинство программ, работающих с мышью, допускают также ввод с клавиатуры.

**Джойстик** — манипулятор в виде укрепленной на шарнире ручки с кнопкой. Употребляется в основном для компьютерных игр.

**Графопостроитель или плоттер** — устройство для вывода чертежей на бумагу. Плоттеры несколько дешевле, чем лазерные принтеры, но скорость печати у них ниже. Плоттеры бывают барабанного типа (работают с рулоном бумаги) и планшетного типа (в них лист бумаги лежит на плоском столе). Как правило, плоттеры используются в системах конструирования (САПР) для вывода чертежей.

**Сканер** — устройство для считывания графической и текстовой информации в компьютер. Сканеры (вместе с соответствующим программным обеспечением) могут вводить в компьютер рисунки, а также распознавать символы, что позволяет быстро вводить напечатанный (а иногда рукописный) текст в компьютер. Сканеры бывают настольные (они обрабатывают весь лист бумаги целиком) и ручные (их надо проводить над нужным рисунком или текстом). При систематическом использовании (например, в издательских системах) необходим настольный сканер, хотя он дороже.

**Дигитайзер** — устройство для "оцифровки" изображений. Позволяет преобразовать изображения в цифровую форму для обработки в компьютере. Используется в системах обработки изображений.

**Музыкальная приставка** — дает возможность исполнять музыку с помощью компьютера. Без этой приставки компьютер может выводить в каждый момент звук только одного тона, что дает такие же музыкальные возможности, как игра на фортепьяно одним пальцем.

**Сетевой адаптер** — дает возможность подключать компьютер в локальную сеть. При этом пользователь может получать доступ к данным, находящимся на других компьютерах.

**Стриммер** — устройство для быстрого сохранения всей информации, находящейся на жестком диске. Стриммер записывает информацию на кассеты с магнитной лентой, похожие на кассеты для бытовых магнитофонов. Обыкновенная емкость стриммера — 60 Мбайт, но бывают стриммеры и большей емкости.

**Графический планшет** — устройство для ввода контурных изображений. Используется, как правило, в системах автоматического конструирования (САПР) для ввода чертежей в компьютер.

# Глава 2. Начальные сведения об операционной системе DOS

### Что такое операционная система

Операционная система — это программа, которая загружается при включении компьютера. Она производит диалог с пользователем, осуществляет управление компьютером, его ресурсами (оперативной памятью, местом на дисках и т.д.), запускает другие (прикладные) программы на выполнение. Операционная система обеспечивает пользователю и прикладным программам удобный способ общения (интерфейс) с устройствами компьютера.

Основная причина необходимости такой программы, как операционная система, состоит в том, что элементарные операции для работы с устройствами компьютера и управления ресурсами компьютера — это операции очень низкого уровня и действия, которые необходимы пользователю и прикладным программам, — на самом деле состоят из нескольких сотен или тысяч таких элементарных операций.

Например, накопитель на магнитных дисках "понимает" только такие элементарные операции, как включить/выключить двигатель дисковода, установить читающие головки на определенный цилиндр, выбрать определенную читающую головку, прочесть информацию с дорожки диска в компьютер и т.д. Поэтому для выполнения даже такого несложного действия, как копирование файла с одной дискеты на другую (файл — это поименованный набор информации на диске или другом машинном носителе), необходимо выполнить

тысячи операций по запуску команд дисководов, проверке их выполнения, поиску и обработке информации в таблицах размещения файлов на дисках и т.д. Задача еще усложняется следующим:

— имеется около десятка форматов дискет, и операционная система должна уметь работать со всеми этими форматами. Для пользователя работа с дискетами различного формата должна осуществляться абсолютно одинаково;

— файл на дискетах занимает определенные участки, причем пользователь не должен ничего знать о том, какие именно. Все функции по обслуживанию таблиц размещения файлов, поиску информации в них, выделению места для файлов на дискетах выполняются операционной системой, и пользователь может ничего не знать о них;

— во время работы программы копирования может возникать несколько десятков различных особых ситуаций, например, сбой при чтении или записи информации, неготовность дисководов к чтению или записи, отсутствие места на дискете для копируемого файла и т.д. Для всех этих ситуаций необходимо предусмотреть соответствующие сообщения и корректирующие действия.

Операционная система скрывает от пользователя эти сложные и ненужные ему подробности и предоставляет ему удобный интерфейс для работы. Она выполняет также различные вспомогательные действия, например, копирование или печать файлов. Кроме того, операционная система осуществляет загрузку в оперативную память всех программ, передает им управление в начале их работы, выполняет различные вспомогательные действия по запросу выполняемых программ и освобождает занимаемую программами оперативную память при их завершении.

Как правило, персональный компьютер IBM PC работает под управлением операционной системы MS DOS фирмы Microsoft Corp. или ее варианта PC DOS, распространяемого фирмой IBM, либо же под управлением появившейся в 1988 году совместимой с MS DOS операционной системы DR DOS фирмы Digital Research. Далее будут описываться эти три операционные системы, причем они будут обозначаться словом DOS.

## Составные части DOS

Операционная система DOS состоит из следующих частей.

**Базовая система ввода-вывода (BIOS)**, находящаяся в постоянной памяти (постоянном запоминающем устройстве, ПЗУ) компьютера. Эта часть операционной системы является "встроенной" в компьютер. Ее назначение состоит в выполнении наиболее простых и универсальных услуг операционной системы, связанных с осуществлением ввода-вывода. Базовая система ввода-вывода содержит также тест функционирования компьютера, проверяющий работу памяти и устройств компьютера при включении его электропитания. Кроме того, базовая система ввода-вывода содержит программу вызова загрузчика операционной системы.

**Загрузчик операционной системы** — это очень короткая программа, находящаяся в первом секторе каждой дискеты с операционной системой DOS. Функция этой программы заключается в считывании в память еще двух модулей операционной системы, которые и завершают процесс загрузки DOS.

На жестком диске (винчестере) загрузчик операционной системы состоит из двух частей. Это связано с тем, что жесткий диск может быть разбит на несколько разделов (логических дисков). Первая часть загрузчика находится в первом секторе жесткого диска, она выбирает, с какого из разделов жесткого диска следует продолжить загрузку. Вторая часть загрузчика находится в первом секторе этого раздела, она считывает в память модули DOS и передает им управление.

**Дисковые файлы IO.SYS MSDOS.SYS** (они могут называться и по-другому, например, IBMBIO.COM и IBMDOS.COM для PC DOS, DRBIOS.SYS и DRDOS.SYS для DR DOS — названия меняются в зависимости от версии операционной системы). Они загружаются в память загрузчиком операционной системы и остаются в памяти компьютера постоянно. Файл IO.SYS представляет собой дополнение к базовой системе ввода-вывода в ПЗУ. Файл MSDOS.SYS реализует основные высокоуровневые услуги DOS.

**Командный процессор DOS** обрабатывает команды, вводимые пользователем. Командный процессор находится в дисковом файле COMMAND.COM на диске, с которого загружается операционная система. Некоторые команды пользователя, например, type, dir или copy, командный процессор выполняет сам. Такие команды называются внутренними. Для выполнения остальных (внешних) команд пользователя командный процессор ищет на дисках программу с соответствующим именем, и если находит ее, то загружает в память и передает ей управление. По окончании работы программы командный процессор удаляет программу из памяти и выводит сообщение о готовности выполнения команд (приглашение DOS).

**Внешние команды DOS** — это программы, поставляемые вместе с операционной системой в виде отдельных файлов. Эти программы выполняют действия обслуживающего характера, например, форматирование дискет, проверку дисков и т.д.

**Драйверы устройств** — это специальные программы, которые дополняют систему ввода-вывода DOS и обеспечивают обслуживание новых устройств или нестандартное использование имеющихся устройств. Например, с помощью драйверов возможна работа с "электронным диском", т.е. частью памяти компьютера, с которой можно работать так же, как с диском. Драйверы загружаются в память компьютера при загрузке операционной системы, их имена указываются в специальном файле CONFIG.SYS. Такая схема облегчает добавление новых устройств и позволяет делать это, не затрагивая системные файлы DOS.

## Начальная загрузка DOS

Начальная загрузка DOS выполняется автоматически в следующих случаях:

— при включении электропитания компьютера;

— при нажатии на клавишу "Reset" на корпусе компьютера (такая клавиша есть не у всех моделей компьютеров);

— при одновременном нажатии клавиш [Ctrl], [Alt] и [Del] на клавиатуре.

Для выполнения начальной загрузки DOS необходимо, чтобы на дисководе A для гибких дисков (первого дисковода для дискет, подсоединенного к компьютеру) была установлена дискета с записанной операционной системой DOS или чтобы компьютер имел жесткий диск (винчестер) с записанной на нем операционной системой DOS. Как правило, на жесткие диски операционная система DOS записывается фирмой — поставщиком компьютеров.

В начале загрузки работают программы проверки оборудования, находящиеся в постоянной памяти компьютера. Если они находят ошибку, то выводят код ошибки на экран. Если ошибка не критическая (т.е. дающая возможность продолжения работы), то пользователю предоставляется возможность продолжить процесс загрузки, нажав клавишу [F1] на клавиатуре. Если же неисправность критическая, то процесс загрузки прекращается. При серьезных ошибках о возникшей ситуации и о выданном коде ошибки следует сообщить специалистам по техническому обслуживанию компьютеров.

После окончания работы программ тестирования оборудования программа начальной загрузки пытается прочесть с дискеты, установленной на дисководе A, программу — загрузчик операционной системы. Если на дисководе A нет дискеты, то загрузка операционной системы будет производиться с жесткого диска (винчестера). Если на дисководе A находится не дискета с операционной системой, а какая-либо другая дискета, то будет выдано сообщение об ошибке:

Non-system or disk error
Replace and strike any key when ready
(Несистемный диск или ошибка на диске.
Замените диск и нажмите любую клавишу)

Следует поставить на дисковод A дискету с операционной системой, если Вы хотите загрузить компьютер с дискеты, либо открыть дверцу дисковода или вынуть дискету из дисковода, если Вы хотите загрузить компьютер с жесткого диска (винчестера). После этого следует нажать любую буквенно-цифровую клавишу, пробел или [Enter] для продолжения процесса загрузки.

После того, как с диска, с которого загружается операционная система (т.е. дискеты или винчестера), прочитана программа — загрузчик операционной системы, эта программа считывает в память компьютера модули операционной системы (для MS DOS — файлы IO.SYS и MSDOS.SYS) и передает им управление.

Далее с того же диска читается файл конфигурации системы CONFIG.SYS и в соответствии с указаниями, содержащимися в файле CONFIG.SYS, загружаются драйверы устройств, и устанавливаются параметры операционной системы. Если файл CONFIG.SYS отсутствует, все параметры устанавливаются по умолчанию.

После этого с диска, с которого загружается операционная система, читается командный процессор (файл COMMAND.COM) и ему передается управление. Командный процессор выполняет командный файл AUTOEXEC.BAT, если этот файл имеется в корневом каталоге диска, с которого загружается операционная система. В файле AUTOEXEC.BAT указывают команды и программы, выполняемые при каждом запуске компьютера. Например, там можно указать запуск программы, обеспечивающей работу с русскими буквами на клавиатуре.

Если файл AUTOEXEC.BAT не найден в корневом каталоге диска, с которого загружается операционная система, то DOS запрашивает у пользователя текущую дату и время.

```
GENOA SUPER EGABIOS, Version 3.00
(C) Copyright GENOA Systems Corp7 1986, 1987
Phoenix 80286 ROM BIOS Version 3.00
Copyright (c) 1985, 1986 Phoenix Technologies Ltd
All Rights Reserved
00640K Base Memory, 00384K Expansion
----Installing MOUSE    Device Driver V5.03 ----
Hard Disk (D) Device Driver Installed.
Alias List:
Keyboard driver Installed. EGA 8x14 font loaded.
AntiVirus Installed. To activate menu, press Alt-4.
20:18 C:\»
```

Рис. 2.1. Пример сообщений при начальной загрузке DOS

После выполнения файла AUTOEXEC.BAT процесс загрузки операционной системы заканчивается. DOS выдает приглашение, показывающее, что она готова к приему команд.

На рис. 2.1 показан пример сообщений, которые выдаются при начальной загрузке DOS. Эти сообщения зависят от модели компьютера, версии операционной системы и содержимого файлов CONFIG.SYS и AUTOEXEC.BAT, поэтому на Вашем компьютере сообщения, выдаваемые при загрузке, могут быть совсем другими.

## Резидентные программы

Этот пункт можно при первом чтении опустить.

Как правило, после окончания работы программы вся занимаемая ею оперативная память освобождается и делается доступной для следующих запускаемых пользователем программ. Однако в операционной системе DOS для программ имеется возможность не освобождать (полностью или частично) по окончании сво-

ей работы занимаемую ими оперативную память. Такие программы называются *резидентными*, или постоянно находящимися в памяти.

При запуске резидентной программы она выполняет какие-то действия, после чего оканчивает свою работу. На экране появляется приглашение DOS, и пользователь может запускать другие программы. Однако часть оперативной памяти компьютера остается занятой резидентной программой.

Иногда пользователь может повторно выдавать команду запуска резидентной программы для установки каких-то режимов ее работы. При этом программа, как правило, устанавливает, что она уже является резидентной и не отбирает больше оперативной памяти у DOS.

Наличие резидентных программ имеет смысл потому, что при первом запуске они указывают операционной системе DOS, что она для выполнения некоторых своих услуг должна вызывать заложенные в этих программах подпрограммы. Данные подпрограммы и располагаются в той части оперативной памяти, которая не освобождается при первом запуске резидентной программы.

Например, резидентная программа может установить собственную подпрограмму для обработки ситуаций нажатия пользователем клавиш на клавиатуре. Такая подпрограмма может проверять, не нажата ли пользователем определенная комбинация клавиш, и если она нажата, то вызывать некоторую подпрограмму, а если нет — передавать управление стандартной подпрограмме DOS для обработки нажатия клавиш. Так работают многие известные резидентные программы, например, SideKick, Norton Guides и др. Несколько более сложно обрабатывают нажатие клавиш драйверы клавиатуры, предназначенные для ввода русских букв с клавиатуры, а также программы, расширяющие возможности клавиатуры, например, SuperKey, SmartKey и др.

Резидентные программы могут использоваться и для некоторых других функций, например, для выполнения каких-то особых действий по управлению устройствами компьютера, разграничению доступа к файлам на компьютере, защите от компьютерного вируса и т.д. Резидентные программы должны занимать большой объем оперативной памяти и весьма нетривиальным образом взаимодействовать с операционной системой DOS, поэтому они пишутся достаточно квалифицированными программистами, как правило, на языках Ассемблер и Си.

# Глава 3. Файлы и каталоги на дисках

## Что такое файл

Информация на магнитных дисках хранится в файлах. Файл — это поименованная область на диске или другом машинном носителе. В файлах могут храниться тексты программ, документы, готовые к выполнению программы и т.д.

Часто файлы разделяют на две категории — текстовые и двоичные. Текстовые файлы предназначены для чтения человеком. Они состоят из строк символов, причем каждая строка оканчивается двумя специальными символами "возврат каретки" (CR) и "новая строка" (LF). При редактировании и просмотре текстовых файлов эти специальные символы, как правило, не видны. В текстовых файлах хранятся тексты программ, командных файлов DOS и т.д. Файлы, не являющиеся текстовыми, называются двоичными.

Текстовый файл, содержащий только символы с кодами до 127 (т.е. не содержащий русских букв и псевдографических символов), называется ASCII-файлом.

## Имена файлов

Каждый файл на диске имеет обозначение, которое состоит из двух частей: имени и расширения (часто имя и расширение вместе также называются именем, как правило, это приводит к путанице). В имени файла может быть от 1 до 8 символов. Расширение начинается с точки, за которой следуют от 1 до 3 символов. Например,

command.com
paper.chi
autoexec.bat

имя расширение

Символы в имени и расширении могут быть обозначены прописными и строчными латинскими буквами, цифрами и символами

-_$#&@!%(){}'-`

Расширение имени файла является необязательным. Оно, как правило, описывает содержание файла, поэтому использование расширения весьма удобно. Многие программы устанавливают расширение имени файла, и по нему Вы можете узнать, какая программа создала файл.

Примеры:

| | |
|---|---|
| .com, .exe | — готовые к выполнению программы; |
| .bat | — командные (Batch) файлы; |
| .chi | — докумены для редактора ChiWriter; |
| .pas | — программы на Паскале; |
| .for | — программы на Фортране; |
| .c | — программы на Си; |
| .asm | — программы на Ассемблере; |
| .bak | — копия файла,делаемая перед его изменением. |

В имени и расширении имени файла прописные и строчные латинские буквы являются эквивалентными, так как DOS переводит все строчные буквы в соответствующие прописные буквы.

следует заметить, что многие программы используют расширение .BAK для копий файла, делаемых перед его изменением. Наличие такой копии позволяет восстановить содержимое файла в случае его ошибочного изменения или удаления. После окончания работы с файлом, когда пользователь правильно внес все изменения в файл, он может уничтожить созданные файлы с расширением .BAK.

## Работа с устройствами

Операционная система DOS позволяет с помощью специальных (зарезервированных) имен осуществлять ввод и вывод информации не только с файлами на дисках, но и с различными устройствами компьютера. При этом работа с этими устройствами происходит так же, как с файлами, только в соответствующей команде необходимо вместо имени файла на диске указать имя устройства.

Имена устройств не могут использоваться в качестве имен файлов. Эти имена таковы:

PRN — принтер;

LPT1—LPT3 — устройства, присоединяемые к параллельным портам 1-3 (обычно это принтеры);

AUX — дополнительное устройство, присоединяемое к асинхронному последовательному порту 1;

COM1—COM3— устройства, присоединяемые к асинхронным последовательным портам 1-3;

CON — при вводе — клавиатура, при выводе — экран;

NUL — "пустое" устройство; все операции звода-вывода для этого устройства игнорируются.

Даже если добавить к этим именам какое-либо расширение, все равно DOS будет воспринимать это как обращение к устройству. Например, обращение к файлу CON.ABC эквивалентно обращению к консоли, т.е. к CON, и поэтому CON.ABC не может быть использовано как имя дискового файла. Однако расширения имени файлов .CON, .AUX, .PRN и .NUL вполне допустимы.

Наиболее часто используются устройства PRN (принтер), CON (при вводе — клавиатура, при выводе — экран) и NUL (пустое устройство). Проиллюстрируем их применение. Для этого, несколько забегая вперед, скажем, что команда

copy имя-файла-1 имя-файла-2

копирует информацию из файла, указанного первым параметром, и создает копию этого файла с именем, указанным во втором параметре. Например, команда copy aaa bbb копирует файл aaa в файл bbb. Тогда, если употребить вместо имени выходного файла (bbb) имя PRN, то информация, которая должна выводиться в выходной файл bbb, будет выводиться на принтер. Иначе говоря, команда

copy aaa prn

копирует файл aaa на принтер. Аналогично, команда

copy aaa con

копирует файл aaa на экран. Если же употребить CON вместо имени входного файла, например в команде

copy con bbb

то ввод информации будет осуществляться с клавиатуры (при этом для разделения строк вводимого файла надо нажимать клавишу [Enter], а для окончания ввода — [F6] и [Enter]).

Устройство NUL работает следующим образом: при чтении с него программе сообщается о конце файла, а при выводе на него информация на самом деле никуда не выводится, но программе, которая делала вывод, сообщается, что вывод произошел успешно.

Например, пусть программа PROG имеет три параметра: первый — имя входного файла, второй — имя выходного файла, а третий — имя файла с сообщениями об ошибках. Если третий файл не нужен (скажем известно, что ошибок нет), то можно вызвать программу так:

PROG имя-входного-файла имя-выходного-файла nul

Другое назначение устройства NUL — устранение ненужного вывода на экран у некоторых программ и команд. Например, команда

copy aaa bbb > nul

делает то же, что и команда copy aaa bbb, т.е. копирует файл aaa в файл bbb, но при этом не выводит на экран сообщение

1 file(s) copied

(Более подробно о значении символа ">" будет рассказано при обсуждении перенаправления ввода-вывода в командах DOS).

## Каталоги

Имена файлов регистрируются на магнитных дисках в каталогах (или директориях). Каталог — это специальное место на диске, в котором хранятся имена файлов, сведения о размерах файлов, времени их последнего обновления, атрибуты (свойства) файлов и т.д. Если в каталоге хранится имя файла, то говорят, что этот файл находится в данном каталоге. На каждом магнитном диске может быть несколько каталогов. В каждом каталоге может быть много файлов, но каждый файл всегда регистрируется только в одном каталоге.

**Подкаталоги и надкаталоги.** Все каталоги (кроме корневого, см. ниже) на самом деле являются файлами специального вида. Каждый каталог имеет имя, и он может быть зарегистрирован в другом каталоге. Если каталог X зарегистрирован в каталоге Y, то говорят, что *X — подкаталог Y*, а *Y — надкаталог* или *родительский каталог* для X.

**Имена каталогов.** Требования к именам каталогов те же, что к именам файлов. Как правило, расширение имени для каталогов не используется.

**Корневой каталог.** На каждом магнитном диске имеется один главный или корневой каталог. В нем регистрируются файлы и подкаталоги (каталоги 1-го уровня). В каталогах 1-го уровня регистрируются файлы и каталоги 2-го уровня и т.д. Получается иерархическая древообразная структура каталогов на магнитном диске.

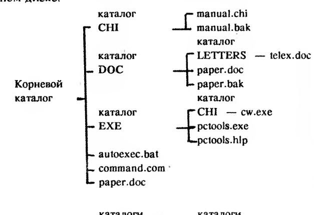



Рис.3.1. Пример файловой системы на магнитном диске

На рис.3.1 в корневом каталоге имеются подкаталоги CHI,DOC и EXE, а также файлы autoexec.bat, command.com и paper.doc. В каталоге DOC имеется подкаталог LETTERS, файлы paper.doc и paper.bak и т.д.

### Текущий каталог

Каталог, с которым в настоящий момент работает пользователь, называется текущнм. Если в команде DOS указать нмя файла, то этот файл будет создаваться илн отыскиваться в текущем каталоге.

Например, команда type выводит содержимое файла на экран. Тогда команда type xxx.doc будет искать файл xxx.doc в текущем каталоге.

Для вывода оглавления текущего каталога необходимо ввести команду dir. Для смены текущего каталога нмеется команда cd.

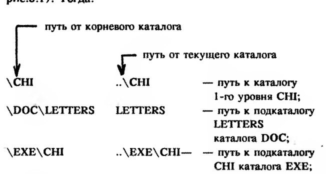
### Указание пути к файлу

Когда вы используете файл не из текущего каталога, необходнмо указать, в каком каталоге этот файл находится. Это делается с помощью указания пути к файлу.

*Путь* — это последовательность из имен каталогов или символов "..", разделенных символом "\". Этот путь задает маршрут от текущего каталога или от корневого каталога диска к тому каталогу, в котором находится нужный файл.

Если путь начинается с снмвола "\", то маршрут вычисляется от корневого каталога диска, нначе — от текущего каталога. Каждое нмя каталога в пути соответствует входу в подкаталог с такнм именем, ".." соответствует входу в надкаталог.

Например, пусть текущий каталог — DOC (см. рис.3.1). Тогда:




### Имена накопителей на дисках (дисководов)

В компьютере обычно имеется несколько накопителей на магнитных дисках (дисководов). Для DOS накопители на магнитных дисках именуются A:,B:,C: и т.д. Например, в компьютере может быть два накопителя на гибком магнитном диске A: и B: и один накопитель на жестком магнитном диске (винчестер) C:.

### Текущий дисковод

Текущий дисковод — это тот дисковод, с которым Вы работаете в настоящее время. DOS по умолчанию ищет все задаваемые пользователем файлы на диске, находящемся на текущем дисководе. Вы можете сменить текущий дисковод
с помощью команд DOS.

### Полное имя файла

Полное имя файла имеет следующий вид:

[дисковод:][путь\]имя-файла

т.е. состоит из путн к каталогу, в котором находится файл, и имени файла, разделенных символом "\", перед которыми может стоять обозначение дисковода.

Если дисковод не указан, то подразумевается текущий дисковод. Если путь не указан, то подразумевается текущнй каталог.

Полное имя файла полностью специфицирует, с каким файлом Вы хотите работать.

Например, пусть на рис. 3.1 изображена файловая система на диске накопителя A:. Текущий каталог на этом накопителе — A:\DOC. Тогда

| | |
|---|---|
| a:paper.doc | — файл paper.doc в текущем каталоге диска на дисководе A:; |
| a:\paper.doc | — файл paper.doc в корневом каталоге диска на дисководе A:; |
| post\telex.doc | — файл telex.doc в подкаталоге POST текущего каталога. |

### Символы * и ?

Во многих командах в именах файлов можно употреблять символы * н ? для указания группы файлов из одного каталога.

Символ * обозначает любое число любых символов в имени файла или в расширенин нмени файла. Символ ? обозначает однн произвольный символ или отсутствие символа в имени файла или в расширении имени файла.

В нменах файлов, содержащих указание на каталог или дисковод, символы * и ? нельзя употреблять в той части именн, которая содержит указание на каталог или днсковод. Например, имя a:\work*.doc допустимо, а имена a:*\paper.doc и *:\work\paper.doc — нет.

П р и м е р ы:

*.bak — все файлы с расширением .bak из текущего каталога;

c*.d* — все файлы с именем, начинающимся с C, и с расширением, начинающимся с D, из текущего каталога;

a:\doc\ABC???.* — все файлы с имеием, начинающимся с ABC и состоящим не более чем из 6 символов.

### Логические диски

В операционной системе DOS можно разделнть жесткий диск на несколько частей, н работать с ними

как с отдельными дисками. Эти части называются ***логическими дисками*** **или** ***разделами жесткого диска***. Каждый логический диск имеет имя (букву), по которому к нему можно обращаться. Например, жесткий диск объемом 40 Мбайт может быть разделен на два логических диска: диск C: объемом 32 Мбайта и диск D: объемом 8 Мбайтов. Пользователь может даже не знать, что эти два диска не являются независимыми физическими устройствами, а расположены на одном жестком диске.

### Электронные диски

Если в компьютере имеется достаточное количество оперативной памяти, то можно отвести часть этой памяти под "электронный диск" (RAM-диск). С этой частью памяти можно работать так же, как с диском. Ввод и вывод информации на "электронный диск" осуществляется гораздо быстрее, чем на обычный диск, поскольку это не связано с физическим перемещением диска и считывающих головок. Однако при выключении питания или перезагрузке компьютера информация, записанная на "электронный диск", пропадает.

Для работы с "электронным диском" требуется запуск специальной программы или включение в файл конфигурации системы CONFIG.SYS программы-драйвера "электронного диска"

## Глава 4. Диалог пользователя с DOS

Диалог пользователя с DOS осуществляется в форме команд. Каждая команда пользователя означает, что DOS должна выполнить то или иное действие, например, напечатать файл или выдать на экран оглавление каталога.

Команда DOS состоит из имени команды и, возможно, параметров, разделенных пробелами. Имя команды DOS и параметры могут набираться как прописными, так и строчными латинскими буквами. Ввод каждой команды заканчивается нажатием клавиши [Enter].

### Приглашение DOS

Когда DOS готова к диалогу с пользователем, она выдает на экран приглашение, например

A> или C:\>

Это означает, что DOS готова к приему команд. Когда пользователь проводит диалог с какой-либо программой, а не с DOS, тогда приглашение DOS отсутствует (впрочем, программа может иметь свое приглашение для ввода команд этой программы).

Приглашение DOS, как правило, содержит информацию о текущем дисководе и о текущем каталоге (см. ниже). Например,

A:\> — дисковод A:, корневой каталог.

C:\EXE> — дисковод C:, каталог \EXE.

Иногда приглашение DOS включает также информацию о текущем времени суток, например

12:59 C:\EXE\SYS>

Вид приглашения можно изменить с помощью команды DOS Prompt.

### Ввод команд

Для ввода команды следует набрать эту команду на клавиатуре и нажать [Enter]. При вводе команд можно пользоваться следующими клавишами для редактирования вводимой команды:

[Del] — удаление текущего символа;

[Backspace] — (стрелка налево над клавишей [Enter]) — стирание предыдущего символа;

[Ins] — включение-выключение режима вставки;

[Esc] — очистка всей командной строки.

Если при начальной загрузке операционной системы была запущена программа DOSEDIT (см.ниже), то можно пользоваться и следующими клавишами:

[→],[←] — (стрелки направо и налево на функциональной части клавиатуры) — перемещение курсора;

[↑] — (стрелка вверх на функциональной части клавиатуры) — выводит в командную строку предыдущую команду.

Более подробно о редактировании вводимых команд рассказано в конце этой главы.

### Запуск и выполнение команд

Любая команда, вводимая пользователем, указывает на необходимость выполнения либо внутренней, либо внешней команды DOS, либо других программ или командных файлов.

Для выполнения внутренней или внешней команды DOS необходимо ввести имя этой команды и ее параметры.

**Поиск выполняемой программы.** Когда пользователь вводит команду, которая не относится к числу внутренних команд DOS, командный процессор ищет программу с именем, указанным в команде. Поиск проводится среди файлов со следующими расширениями:

.COM — программные файлы;

.EXE — программные файлы (в другом формате);

.BAT — пакетные командные файлы.

Поиск выполняется в том порядке, в котором эти расширения перечислены выше. Если пользователь не указал, в каком каталоге следует искать программу, то поиск производится в текущем каталоге и в каталогах, заданных командой DOS Path.

Если нужная программа в этих каталогах не найдена, то на экран выводится сообщение

Bad command or file name

(Неверное имя команды или программы)

Например, если пользователь ввел команду PROG, то командный процессор будет искать в текущем каталоге и в каталогах, заданных командой DOS Path, файл с именем PROG.COM или PROG.EXE, или PROG.BAT. Если найденный файл имеет расширение .COM или .EXE, то командный процессор выполняет

загрузку этого файла в память и передает ему управление, чтобы этот файл мог выполнить свою работу. Если найденный файл имеет расширение .BAT, т.е. является командным файлом (т.е. файлом пакетной обработки), то он содержит в формате текстового файла (в коде ASCII) последовательность команд, которые должны выполняться так, как будто они вводятся с клавиатуры.

**Явное указание каталога в команде.** В версиях DOS, начиная с 3.00, а также в DR DOS, в командах можно явно указывать имя каталога, в котором надо искать соответствующую программу. Для этого следует в качестве имени команды использовать полное имя файла, включающее путь к каталогу, в котором находится нужная программа или командный файл. Расширение имени файла (т.е. .COM, .EXE и .BAT) можно не указывать.

Например, пусть текущий каталог — C:\DOC\WORK, а надо выполнить программу HYPHEN, находящуюся в каталоге C:\DOC\PROG, и указать параметры программы PAPER.DOC /P. Тогда для выполнения этой программы необходимо выполнить команду:

c:\doc\prog\hyphen paper.doc /p

или

..\prog\hyphen paper.doc /p

## Действия при "зависании" компьютера или неправильной работе программ

Иногда выполняемая программа начинает работать неправильно или же не реагирует на нажатие клавиш и т.д. В этом случае выполнение программы следует прекратить. Это делается так:

— сначала надо одновременно нажать на клавиши [Ctrl] и [Break], чтобы прекратить выполнение программы или вывести ее из состояния "зависания";

— если это не помогает, то надо перезагрузить DOS. Для этого следует одновременно нажать на клавиши [Ctrl], [Alt] и [Del];

— если при нажатии [Ctrl], [Alt] и [Del] компьютер не перезагружается, то следует нажать клавишу "Reset" на корпусе компьютера;

— если в Вашем компьютере нет клавиши "Reset", то надо выключить компьютер, а затем включить его.

Вы можете прекратить выполнение любой команды DOS, нажав комбинацию клавиш [Ctrl—C] или [Ctrl—Break] (как указывалось в части 1, нажать [Ctrl—C] означает нажать клавишу [Ctrl], и, не отпуская ее, нажать клавишу "C").

Заметим, что прикладные программы не обязаны (хотя и могут) реагировать на нажатие [Ctrl—C] или [Ctrl—Break].

## Приостановка вывода на экран

Если команда DOS выдает слишком много информации на экран, можно воспользоваться комбинацией клавиш [Ctrl—S] для приостановки выдачи. Повторное нажатие [Ctrl—S] возобновит выдачу.

Для других программ выдачу информации на экран можно приостановить нажатием комбинации клавиш [Ctrl—NumLock]. Для продолжения выполнения надо нажать любую клавишу.

## Пауза при выполнении команд

Если при выполнении команды DOS на экране появляется сообщение

Strike and key when ready

(Нажмите любую клавишу, когда будете готовы)

или

Strike any key to continue

(Для продолжения нажмите любую клавишу)

то для продолжения работы следует нажать любую буквенно-цифровую клавишу, пробел или [Enter].

## Редактирование вводимых команд DOS

DOS предоставляет некоторые возможности по редактированию вводимых команд во время их набора, однако они весьма ограничены и неудобны. Поэтому чаще всего пользователи предпочитают запускать какую-либо специальную резидентную программу, которая позволяет выводить в командную строку одну из нескольких последних введенных команд и редактировать команду в командной строке более удобным образом. Описание возможностей одной из таких программ — программы DOSEDIT — приводится ниже.

Для установки программы DOSEDIT необходимо скопировать программу DOSEDIT. COM на компьютер (или на системную дискету), и вставить в файл AUTOEXEC.BAT команду вызова программы DOSEDIT. В простейшем случае эта команда выглядит так: DOSEDIT. После вызова DOSEDIT устанавливается резидентно, и предоставляемые ею возможности доступны до перезагрузки компьютера. После запуска программы DOSEDIT командную строку DOS можно редактировать следующим образом.

**Перемещение курсора:** [→] — перемещение курсора вправо; [←] — перемещение курсора влево; [Ctrl—→] — перемещение курсора вправо на слово; [Ctrl—←] — перемещение курсора влево на слово; [LeftShift—Tab] — перемещение курсора влево к следующей позиции табуляции; [RightShift—Tab] — перемещение курсора вправо к следующей позиции табуляции; [Home] — перемещение курсора к началу строки; [End] — перемещение курсора к концу строки.

**Удаление символов:** [Del] — удаление символа под курсором; [Backspace] — удаление символа слева от курсора; [Esc] — очистка командной строки; [Ctrl—Home] — очистка командной строки от текущей позиции курсора до начала строки; [Ctrl—End] — очистка командной строки от текущей позиции курсора до конца.

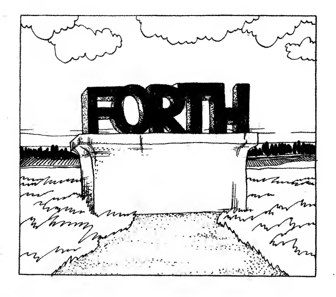
**Прочие команды:** [Ins] — включение и выключение режима вставки. В режиме вставки курсор увеличивается в толщине. При нажатии клавиш [Enter] и

[Esc] режим вставки выключается; [Ctrl—Z] — ввод в командную строку символа конца файла (символа с кодом 26).

**Вызов в командную строку ранее введенных команд.** Программа DOSEDIT хранит несколько последних введенных пользователем команд и может выводить их в командную строку DOS. Это позволяет легко повторить одну из последних команд, сделав при необходимости в ней нужные изменения. Команды хранятся в виде кольцевого стека. Для вызова в командную строку ранее введенных команд и управления стеком команд можно использовать следующие клавиши:

[↑] — вызов предыдущей команды;
[↓] — вызов следующей команды;
[Ctrl—PgUp] — очистка стека;
[Ctrl—PgDn] — очистка из стека текущей команды (т.е. команды, изображенной в командной строке).

*(Продолжение следует)*




# ЯЗЫК ФОРТ

## НЕМНОГО ИСТОРИИ

Приступая к созданию Форта, Чарлз Мур задался целью не просто создать новый язык, а найти способ повышения собственной производительности. "Традиционные языки программирования не обеспечивали мощности, легкости и гибкости, которые могли бы меня устроить, — объяснял он позднее.— Я подсчитал, что за 40 лет очень хороший программист может написать 40 программ, мне же это показалось недостаточным. В мире столько вещей, которые нужно сделать, что мне требовался инструмент, способный помочь в этом".

Чарльз Мур родился в штате Мичиган. В 1960 г. он окончил Массачусетский Технологический Институт, где его специализацией была физика. В начале 60-х годов Мур начал разрабатывать элементы языка Форт, создавая программы для Станфордского линейного ускорителя в Калифорнии. Одной из особенностей Форта являлась компактность; система его обозначений была настолько лаконична, что некоторые ключевые слова представляли собой просто знаки пунктуации. Еще одно свойство, необычное для других языков, — легкая расширяемость: программист мог без труда определять новые ключевые слова или команды в терминах уже существующих, настраивая набор команд языка для любого приложения.

В конце 60-х годов, когда Мур выполнял расчеты для одной нью-йоркской частной компании, его идеи воплотились в том, что с полным правом можно было назвать языком программирования. Вначале Мур хотел дать своему детищу название FOURTH (четвертый), поскольку, на его взгляд, созданный им язык превосходил по своей мощности язык компьютеров третьего поколения. (По словам Мура, в своем окончательном виде новый язык позволил ему работать в 10 раз продуктивнее.) Однако машина IBM-1130, на которой Мур в то время работал, ограничивала длину идентификаторов пятью литерами, поэтому название языка пришлось сократить до FORTH. (Позднее он охарактеризовал это сокращение как "утонченную игру слов": FORTH в переводе с английского обозначает "вперед".)

Форт стал активно применяться в начале 70-х годов, когда Мур работал в Национальной радиоастрономической обсерватории в Аризоне. В сотрудничестве с Элизабет Разер, ведавшей программным обеспечением обсерватории, он использовал Форт при написании серии программ для мини-компьютеров, которые, в частности, управляли в реальном масштабе времени системой наведения 11-метрового телескопа обсерватории Кит-Пик. Программы и сама система оказались настолько удачными, что в 1973 г. Мур, Разер и их руководитель Нед Конклин решили создать свою компанию, назвав ее "FORTH Incorporated". Компания занималась продажей систем, пригодных не только для обсерваторий, но и для других специальных приложений, где требовалось управление в реальном масштабе времени. Одна из таких систем — система автоматического управления видеокамерами — была установлена на подводном аппарате, участвовавшем в 1985 г. в поисках затонувшего "Титаника".

Рынок персональных компьютеров особого интереса для компании не представлял; тем не менее, Форт вызвал ажиотаж среди программистов-любителей. В числе первых был Ким Харрис, молодой инженер из Кремниевой долины. На одном из семинаров компании он стал свидетелем того, как демонстратор за 15 мин составил простую программу для исполнения компьютерной музыки, что привело его в неописуемый восторг: Харрис знал квалифицированного любителя, который трудился больше года, чтобы получить подобную программу на языке ассемблера. "Это было подобно чуду, — вспоминал он, — и я увидел его собственными глазами".

В 1977 г. Харрис создал инициативную группу под названием FIG (FORTH Interest Group), которая занялась разработкой дешевой Форт-системы для любителей. Помимо него в FIG входили еще четверо молодых энтузиастов. Пятеро основателей и семеро их добровольных помощников, получив из обсерватории первую широкодоступную версию интерпретатора языка Форт, трудились более полугода ночами и в свободное от основной работы время над созданием упрощенного интерпретатора для персональных компьютеров. Так появилась система FIG-FORTH, которая стоила всего около 20 долларов. В дальнейшем были разработаны и другие версии Форт для микрокомпьютеров, но любители отдавали предпочтение FIG-FORTH. Сама группа тем временем насчитывала в своем составе уже около 4000 человек из разных концов страны.

Форт малопонятен тому, кто никогда им не пользовался.

Однако его сторонники утверждают, что этот язык "усиливает их власть" над компьютером (подобно тому, как ручная передача подчиняет автомобиль водителю). "Язык, подобный Форту, — это рай для хакеров", — заметил один из энтузиастов.

## Форт и встроенные системы

Имеется ряд реализаций языка (например, PolyFORTH), вполне применимых к микропроцессорам. PolyFORTH имеет средства для мультипрограммирования и представляет собой замкнутую систему с дисковой поддержкой этапа создания и отладки, а также с обеспечением, требуемым для выполнения программы в ППЗУ объектной системы.

Что же может привлекать в Форте инженера-разработчика? Прежде всего, наверное, два его необычных свойства: полный доступ к особенностям ЭВМ и уникальная возможность использования ассемблера в диалоговом режиме. Кроме того, Форт очень компактен: для вызова любой операции языка, независимо от ее сложности, требуется всего два байта. Чем больше размер программы, тем больше экономия, чему способствует иерархическая структура прикладной задачи и, соответственно, программы. Модульность кода приводит к отсутствию дублирования отдельных его кусков. Если ассемблерная программа занимает 4 Кбайта, то соответствующая программа на Форте будет примерно такой же. Если ассемблерная программа занимает 8 Кбайт, размеры программы на Форте (при условии, что она грамотно написана) составят 6.5—7 Кбайт.

Главное преимущество Форта как языка программирования — его гибкость, позволяющая оптимизировать программный продукт по трем критериям: времени написания, времени выполнения и требуемому для работы объему памяти.

*В.Форсюк*

Материал получен при содействии редакции бюллетеня "Софтпанорама".




# НОВОСТИ

Несмотря на ряд объявлений о скором появлении версии Microsoft Windows с вводом информации "электронным пером", она появится не ранее, чем через год. Представители фирмы также не подтвердили слухи об активно ведущейся разработке 32-разрядной версии Windows.

Как сообщил представитель Microsoft Конни Балмер, версия с вводом рукописной информации будет представлять собой расширение Windows 3.1 (планируемой улучшенной версии Windows 3.0). "Мы сообщили об этом ряду представителей прессы и экспертам, продемонстрировав при этом альфа-версию". (Альфа-версия — тестовая версия, проверяемая специалистами фирмы, а бета-версия — почти законченный вариант, передаваемый для испытаний независимым экспертам).

Система получила название Pen Windows. Ее особенностью является возможность ввода информации и команд не только с помощью клавиатуры и мыши, но и путем распознавания обычного рукописного текста. На сегодняшний день Pen Windows способна воспринимать только строчные печатные буквы. Pen Windows, по сообщениям, будет понимать и "редакторские жесты", такие, как зачеркивание части текста для его удаления или обведение в рамку для выделения. Чтобы обеспечить выполнение этих команд, к нынешним 550 встроенным функциям Windows 3.0 добавятся еще около 35.

*Newsbytes News Network, 22 Jan, 1991*

С 14 по 17 мая в Киеве состоится конференция и выставка современных технологий Computer-Aided Software Engineering (CASE). Она будет организована киевской фирмой Технософт (телефон (044) 266-0079) при поддержке ЮНЕСКО.

В Москве на ВДНХ с 6 по 12 апреля 1991 г. будет проходить вторая в нашей стране выставка Comtec. Как и в прошлом году, ее организацией занимается СП Crocus International и американская корпорация ComputerLand. Желающие принять участие в выставке могут получить необходимую информацию по телефону (095) 181-9717.

*Newsbytes News Network, 22 Jan, 1991*

Последние бета-версии MS-DOS 5.0 представлены на рассмотрение независимых экспертов с заверениями в том, что это последняя версия, и выпуск продукта в свет ожидается в марте этого года.

Уже сейчас известно, что операционная система будет включать в себя ряд утилит, которые ранее выпускались независимыми производителями, — например, средства восстановления информации на отформатированных дисках, являющиеся одним из основных достоинств пакета PCTools фирмы Central Point Software.

Между тем, Digital Research — компания, уже выпускающая свою версию DOS (DR-DOS 5), — заявила о своей готовности выпустить на рынок DR-DOS 6, которая помимо усовершенствований MS-DOS включает элементы, которых в ней нет.

*Newsbytes News Network, 21 Jan, 1991*

Как сообщается, новая версия издательского пакета Pagemaker для персональных компьютеров фирмы IBM уже практически готова.

После того как версии продукта для PC и Macintosh впервые были продемонстрированы прессе, представители фирмы Aldus объясняли, что выпуск в свет PC-версии откладывается в связи с задержкой Windows

3.0. Но вот Windows уже в продаже, а долгожданной PageMaker 4.0 все еще нет. Теперь компания заявляет, что за такое короткое время внести изменения и добавления, необходимые для конкуренции с последней версией пакета Ventura Publisher было просто невозможно. Ожидается, что версия для PC, которую намечено выпустить в феврале. будет отличаться от программы для Mac.

Среди возможностей версии — с нетерпением ожидаемые пользователями программы типографского цветоделения, создания индексов, взаимосвязь нескольких файлов и встроенный текстовый процессор. В Австралии новая версия будет продаваться за 1295 австралийских долларов вместо 1595 долл. — стоимости нынешней версии 3.01.

*Newsbytes News Network, 21 Jan, 1991*

Продажа двух миллионов копий пакета Windows 3.0 позволила фирме Microsoft увеличить свои доходы на 53%. Компания продолжает шокировать Уолл-стрит своими финансовыми успехами. По сообщению фирмы, доход за четвертый квартал 1990 г. составил 460,5 млн. долл. при доходе за аналогичный период предыдущего года "всего" в 300,4 млн.

Windows 3.0 поступил в продажу в мае 1990 г. Пакет, наконец, предоставил пользователям интерфейс, который по простоте может сравниться с Macintosh. С тех пор появилось более 1000 программ, специально написанных или доработанных для работы в среде Windows.

*Newsbytes News Network, 21 Jan, 1991*

Французский национальный центр космических исследований и автомобильная компания Пежо приобрели компьютеры Cray общей стоимостью 15-25 млн. долл.

*Newsbytes News Network, 23 Jan, 1991*

Фирма Microsoft выпустила версию 3.0 своей программы работы с таблицами Excel. Она содержит ряд новых элементов, которые позволят серьезно потеснить конкурентов. Новая версия была одновременно продемонстрирована для работы в среде MS-DOS, OS/2 Presentation Manager и Macintosh.

Среди более чем 100 новых режимов появились такие, как возможность получения суммы ячеек в строке или столбце с помощью кнопки Autosum, подсказки для пользователей Lotus 1-2-3 и возможность построения трехмерных графиков, которые можно вращать прямо на экране.

Кроме того, полностью поддерживается новый протокол связи и объединения объектов фирмы Microsoft (Object Linking and Embedding (OLE)), посредством которого прикладные программы в среде Windows могут вызывать друг друга и передавать необходимые данные. Второй и последний на сегодняшний день пакет, поддерживающий этот протокол, — это PowerPoint фирмы Microsoft, предназначенный для создания цветных графических изображений. Все прочие программы компании используют динамический обмен данными Dynamic Data Exchange (DDE) — более ранний и менее изысканный подход к интеграции.

"Microsoft была поставлена перед фактом, что Lotus остается самой популярной табличной программой, а среди текстовых процессоров первенство держит WordPerfect. Однако теперь, после появления Word for Windows, совместно работающих Excel и PowerPoint, фирме представился реальный шанс коренным образом изменить эту тенденцию", — сказал Джон Стаэл, президент фирмы Store Board Computer Intelligence.

Еще одним важным моментом является то, что новая программа работает и в среде Windows, поддерживая интерфейс New-Wave фирмы Hewlett-Packard. И хотя Lotus имеет графическую версию своей табличной программы, Windows в ней не поддерживается.

Среди новых режимов — средства рисования и возможность использования графических изображений из Windows Clipboard. Пользователь может выбрать 68 различных типов графиков.

Версия для Windows 3.0 продается уже сейчас, а поставки двух других версий начнутся в первой половине 1991 г. В США пакет будет стоить 495 долл. Зарегистрированные пользователи Windows смогут приобрести пакет по цене 129 долл., а те, кто купил Excel for Windows 2.1 в период с 7 декабря 1990 г. по 16 апреля 1991 г., — всего за 50 долл. В этом году ожидается также появление французской, немецкой, шведской, итальянской, испанской и датской версий продукта. Новый режим International Macro Sheets позволит создавать макрокоманды, которые одинаково будут работать во всех версиях, независимо от страны-производителя. Продажа английской версии начинается в 19 странах мира.

*Newsbytes News Network, 10 Jan, 1991*

Компания XTree выпустила новую версию своей программы работы с дисками XTreeGold 2.0. Не отказываясь от своей программы по охране окружающей среды, фирма за каждую присланную ей регистрационную карточку пользователя обещает посадить одно дерево.

Цена пакета — 149 долл., для владельцев предыдущих версий — 35 долл. Программа поставляется на дискетах размером 5.25 и 3.5 дюйма и работает на любых машинах PC XT, AT, PS/2 с ОЗУ емкостью 256 Кбайт и операционной системой, начиная с DOS 3.1.

*Newsbytes News Network, 7 Jan, 1991*

*К.Чащин*

*Советско-американское предприятие «Соваминко»*

*Рекламно-издательское агентство «КомпьютерПресс»*

*Принимает заказы на журнал «КомпьютерПресс» и производит отправку наложенным платежом.*

*Заказ высылается по адресу: 191186, Ленинград, Невский проспект, 28 магазин №1 «Дом книги»*

## ЗАКАЗ

**От кого** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

**Адрес** . . . . . . . . . . (почтовый индекс указывать обязательно) . . . . . . . . . .

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

**Номера выпусков** . . . . . . . . . . . . **Количество экземпляров** . . . . . . .

---

**На обратной стороне этой страницы помещен бланк заказа на сборник «КомпьютерПресс»**
**Вы можете его вырезать и, заполнив, отправить в конверте по адресу:**
**113093, Москва, а/я 37.**

**Подписка на 1991 г. принимается до 1 апреля 1991 г. Число экземпляров — без ограничений.**

**Вы можете выписать журнал на полгода или на год. Стоимость годовой подписки — 48 рублей, полугодовой — 24 рубля.**

**Деньги следует перечислить на расчетный счет агентства “КомпьютерПресс“.**

*Банковские реквизиты:*
**получатель: Автобанк (для зачисления на счет №345708)**
**расчетный счет получателя: №161202**
**банк получателя: ЦОУ при Госбанке СССР. МФО №299112.**
**Копию платежного документа необходимо приложить к бланку заказа.**

**Без одновременной оплаты подписной стоимости заказ не принимается. Издания агентства “КомпьютерПресс“ наложенным платежом не высылаются.**

## Образец заполнения платежного поручения для предприятий и организаций

**ПЛАТЕЖНОЕ ПОРУЧЕНИЕ №** ____ 0401002

________ 199 г.

| | ДЕБЕТ | Сумма |
|---|---|---|
| Плательщик | | |
| Банк плательщика в г. | сч. № | *48-00* |
| Получатель *Автобанк (для зачисления на счет 345708)* | КРЕДИТ | |
| | сч. № 161202 | |
| Банк получателя *ЦОУ при Госбанке СССР в г. Москве МФО № 299112* | | |
| почтой-телеграфом (нужное подчеркнуть) | | пеня за из М Р. |
| Сумма прописью | | сумма с пеней |
| | | Вид опер. |
| Назначение платежа *Подписка на сборник "КомпьютерПресс"* | | Назн. плат. |
| | | Срок плат. |
| | | Очер. плат |
| | | № гр. банка |

М.П. Подписи клиента

Проведено банком
199 г.
Подписи банка

---

# ЗАКАЗ

**От кого** ________________________________

**Адрес** ________________________________

(ПОЧТОВЫЙ ИНДЕКС УКАЗЫВАТЬ ОБЯЗАТЕЛЬНО)

**Прошу оформить подписку на 1991 год**

**Подписная плата в сумме** ____________________ **перечислена**

**платежным поручением (почтовым переводом) №** ______ **от** __________ **1991 г.**

**(Копия платежного документа прилагается)**

Цена 3р15